HX-5509A 开发板使用手册

前 言

本用户指南是 TMS320VC5509A 开发板的硬件使用说明书，详细描述了该板的硬件构成、原理，以及它的使用方法和编程指导。

每块板子都经过精密焊接、严格测试的过程。从原材料的购买、原材料的老化选择、半成品的制作、成品检测、成品老化。保证了每个板子过硬的质量，请放心使用

用户在使用板子之前，请详细阅读本使用手册。对于由于未阅读本手册而造成的硬件损坏，概不负责退换。

本公司保证所生产制造的产品均经过严格的品质认证流程，同时保证在出厂一年内，如有发现产品在生产过程中产生的瑕疵或零件故障，本公司负责免费维修。但如果用户自行更改电路、功能、或自行维修本产品、更换零件或擦伤、损坏本产品等情况，本公司不提供免费保修服务，视实际情况酌情收取维修、零件费用。如未按规定操作而发生一场情况（带电插拔外扩设备等造成的器件损坏），本公司恕不提供免费保修服务。所有软件产品、用户手册及其它资料享受两年内免费升级，请关注本公司官方网站上的更新公告。

本保证不包括本产品的附属设备（稳压电源、串口线、并口线、USB 线、双绞线、资料光盘）等。

本说明如有错误，敬请指正。

参考资料：

- TMS320C55x Technical Overview
- TMS320C55x DSP CPU Programmer's Reference Supplement (Rev. C)

第一章 概述

1.1 特点

□ 采用 TMS320VC5509A 200MHz

◆片上存储器：

SRAM： 128K X 16 位

ROM： 32K X 16 位

◆片上外设：

◆20 位定时器： 2 路

◆McBSP： 3 通道

◆MMC/SD 接口： 2 通道

◆ADC： 2 通道，10 位，21.5kHz，0～3.3V

◆实时时钟 RTC

◆看门狗电路

◆IIC 总线

◆外扩 SDRAM，配置为 4M X 16 位

◆外扩 MMC/SD 卡接口

◆AC97 标准的 Audio 音频接口

◆外扩 USB2.0 全速 USB 从接口

◆外扩 10M 以太网接口

◆由 CPLD 检测的按键输入

◆由 VC5509 的 GPIO 驱动的 8 个 LED 指示灯

◆完备的总线扩展

1.2 概述

在鸿翔电子 HX-5509A 开发板是为学习、评估 TI 的 TMS320VC5509A 而开发的，主要包含两部分：硬件模板和相应的测试软件。

在鸿翔电子 HX-5509A 开发板上集成了 DSP、SDRAM、Codec、USB、MMC/SD、Ethernet 等接口外设以及开放给用户的 DSP 总线扩展。这样使其能够应用在语音处理及其它相关领域。

相应的测试软件包括：

CPU 看门狗实验

LED 跑马灯实验

CPU Timer 定时器实验

实时时钟实验

扩展 SDRAM 读写实验

扩展 FLASH 读写实验

键盘扫描实验

外部中断输入实验

AIC23 播音实验

LCD 显示实验

串口通信实验

USB2.0 通信实验

网络通信实验

MMC/SD 卡通信实验

1.3 技术指标

主处理器：TMS320VC5509A，主频 200MHz

SDRAM：4M X 16 位，72MHz

Codec：双声道、立体声输入/输出，最高采样率 96kHz

USB：符合 USB1.1 规范，最高速度为 12Mb/s

工作温度：0°C～70°C

第二章 TMS320VC5509A 开发板介绍

2.1 TMS320VC5509

数据位数：16 位定点 DSP

最高主频：200MHz

供电：1.6V 内核，3.3VI/O

结构：哈佛结构（程序和数据分开）

2.2 时钟

VC5509A 有两个外部时钟输入：

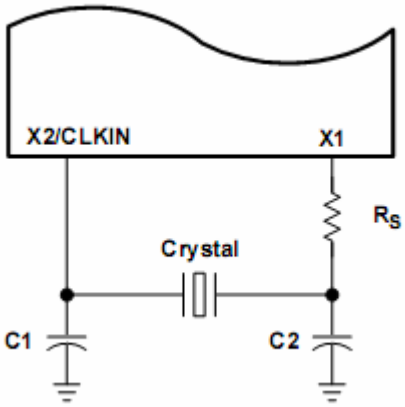
系统时钟：为 CPU 及片内外设提供时钟信号

实时时钟：为 RTC 提供时钟信号，可于系统断电后通过电池供电工作

关于时钟部分请参阅《TMS320VC5509A Data Sheet》。

2.2.1 系统时钟

VC5509 内涵振荡电路，当使用内部振荡电路时，外部镜头的频率范围为 5MHZ ---20MHZ；而使用外部时钟输入时，注意 X2/CLKIN 为时钟输入，而 X1 悬空。



FREQUENCY RANGE (MHz)	MAX ESR (Ω)	TYP C_{LOAD} (pF)	MAX C_{SHUNT} (pF)	R_S (Ω)
20–15	20	10	7	0
15–12	30	16	7	0
12–10	40	16	7	100
10–8	60	18	7	470
8–6	80	18	7	1.5k
6–5	80	18	7	2.2k

由于 VC5509 内部 USB 接口需要一个 48MHZ 的时钟输入，因而在应选时钟输入或晶体的频率为 48 的倍数，这样通过 DPLL 可以实现 48MHZ 时钟输出给 USB 使用。

本开发板采用 12MHZ 晶体为系统提供时钟，对其进行 12 倍频产生 144MHZ CPU 主时

钟，4 倍频产生 USB 所需的 48MHZ 时钟。

VC5509 内部包含一个数字锁相环（DPLL），它可以通过时钟模式寄存器 CLKMD 的 PLL ENABLE 位来使能与禁用。

□ 当 PLL 被禁用时，时钟输出可以作为时钟输入或是其的二分频或四分频。这样的工作方式可以降低功耗。

□ 当 PLL 使能时，可对输入时钟进行适当倍频或分频，就可以获得何时的时钟频率输出。但设置 PLL ENABLE 位，并且当上一个锁相过程已经结束时，VC5509 进入锁相过程。

数字锁相工作过程如下：

下图说明了 VC5509 数字锁相环工作的几个过程状态（A--E）。当时钟模式寄存器 CLKMD 被软件装入或是系统复位时，当 CLKMD 使能 PLL，锁相过程从 A 状态开始，当禁止 PLL 时，时钟产生器进入 D 状态。

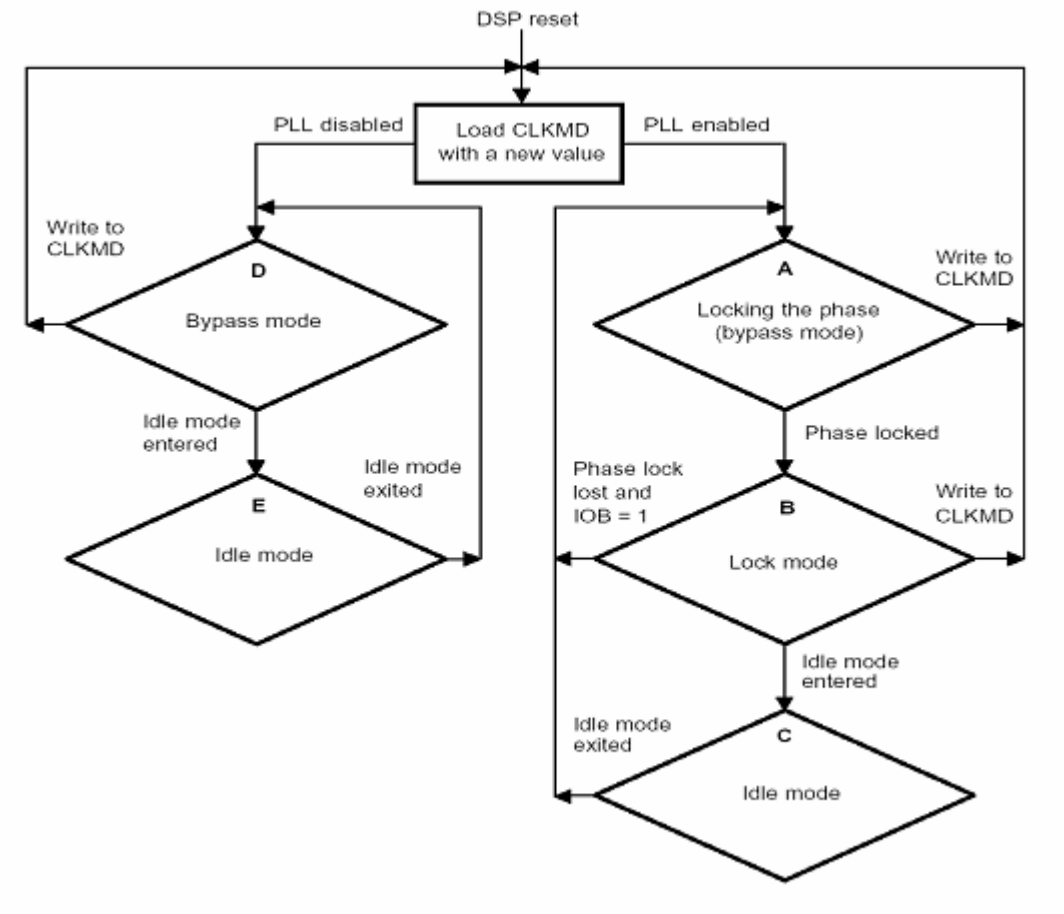
A:已锁相状态。时钟发生器进入屏蔽 PLL 状态。此时输出信号已经与输入信号已经稳定锁相；时钟输出是由 PLL DIV 与 PLL MULT 两位来决定的。此时若重新写入 CLKMD，可以进入下一次锁相过程，即进入 B 状态。

B：锁相状态。如果 CLKMD 寄存器的 IOB 位为 1，锁相过程结束。进入 C 状态；如果 IOB 为 0，则进入 A 状态，重新进行锁定。

C：空闲状态。如果 IDLE 状态存在，IDLE 指令可以将时钟发生器置为 IDLE 状态。当时钟产生器正确从 IDLE 状态退出时，时钟发生器重新启动。

D：PLL 被禁止，时钟输出由 BYPASS DIV 位来控制，通过写 CLKMD 可以改变时钟发生器的状态。

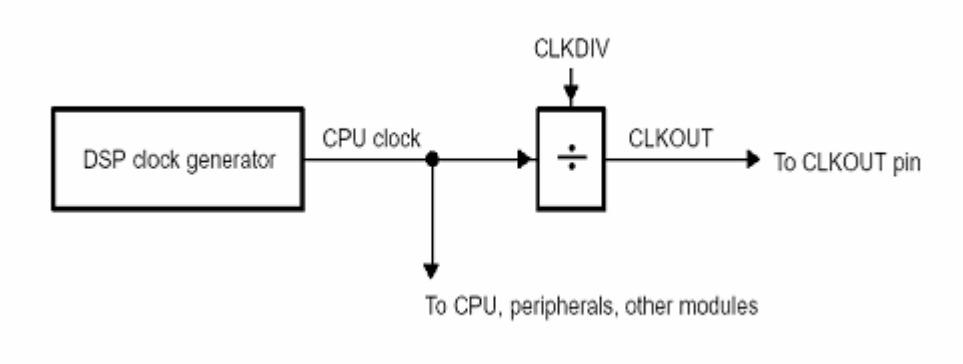
E：空闲状态（从 PLL 禁止状态进入）



系统时钟控制寄存器 CLKMD 的说明如下：

CLKMD Bit Field(s)	Role In The Lock Mode
PLL ENABLE	Allows you to switch to the bypass mode (disable the PLL)
PLL MULT and PLL DIV	Determine how the input clock frequency is modified (if at all) to produce the output clock frequency
IAI	Determines whether the PLL returns to the beginning of the phase-locking sequence when the clock generator exits its idle mode
BREAKLN	Indicates when the phase lock has been broken
IOB	Determines whether the PLL will reacquire a lost phase lock
LOCK	Is 1 in the lock mode

系统时钟与外设及 CLKOUT 的关系如下：



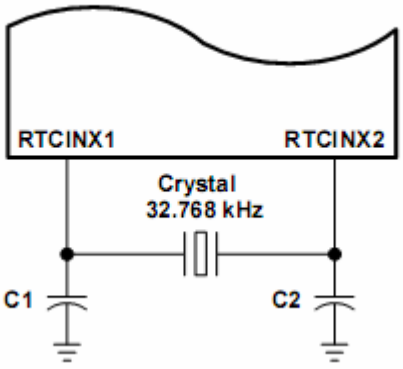
CLKDIV 的设置如下：

CLKDIV	Frequency of CLKOUT
000b	1/1 × CPU clock frequency
001b	1/2 × CPU clock frequency
010b	1/3 × CPU clock frequency
011b	1/4 × CPU clock frequency
100b	1/5 × CPU clock frequency
101b	1/6 × CPU clock frequency
110b	1/7 × CPU clock frequency
111b	1/8 × CPU clock frequency

2.2.2 RTC 时钟：

RTC 需要 32.768KHZ 的晶体连接到 RTCINX1 与 RTCINX2 的输入管脚上。当使用晶振输入时，时钟输入连接到 RTCINX1 上，而 RTCINX2 悬空。当此功能未用时，RTCINX1 接地，RTCINX2 悬空，可用来降低功耗。

本开发板采用 32.768KHZ 晶体为 RTC 提供时钟信号，具体电路如下：



2.3 存储空间

VC5509A 支持统一编址的存储空间，但其 PGE 与 GHH 两种封装所能访问的空间是不同的，区别主要是 GHH 封装共有 21 个地址线，所以它每个片选所能访问的异步空间为 1M X 16 位，而 PGE 封装的只有 14 个地址线，所以它每个片选所能访问的异步空间为 8K X 16 位。

2.3.1 片上存储体

32K X 16 位单周期访问 DRAM，程序/数据均可访问，分为 8 块，每块大小为 4K X 16 位

96K X 16 位单周期访问 SRAM，程序/数据均可访问，分为 24 块，每块大小为 4K X 16 位

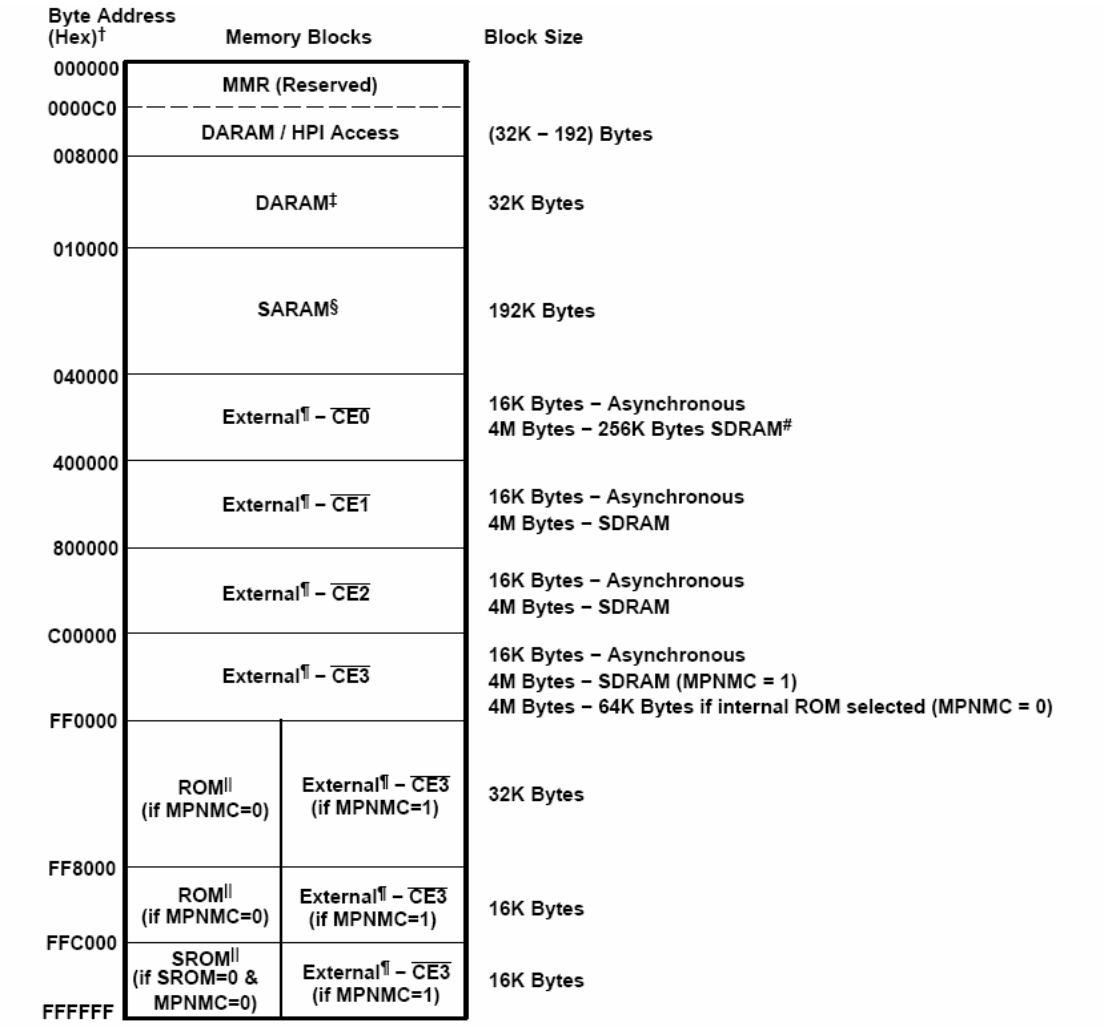
32K X 16 位 1 周期等待 ROM，已经固化了 BootLoader 程序，用于上电引导，用户无法使用

2.3.2 片外存储空间

片外存储空间的访问通过 EMIF（External Memory Interface）接口来完成，VC5509A 片外有 4 个空间（对应 4 个片选信号），支持的存储体类型包括异步 SRAM、FLASH 和 SDRAM。当采用 SDRAM 时，最大可访问空间为 8M X 16 位（片外 4 个空间全用）；当连接 SRAM、FLASH 时，PGE 封装每个片选信号可访问的空间为 8K X 16 位，总共为 32K X 16 位；GHH 封装每个片选信号可访问的空间为 1M X 16 位，总共为 4M X 16 位。

关于 EMIF 的详细说明，参阅《TMS320VC5509 DSP External Memory Interface (EMIF) Reference Guide》。

下面给出 PGE 封装的存储空间分配图。（见《TMS320VC5509A Data Sheet》）



明伟 TMS320VC5509A 开发板外扩一片 16 位的 SDRAM，用/CE0 选通，容量为 4M X 16 位，寻址占用/CE0 和/CE1 两个存储空间。最大可配置为 8M*16 位，占用 CE0—CE3 全部四个片外存储空间。

片选 CE0---CE3 引至总线扩展器上，供用户自行连接设备。

2.4 增强 HPI 与 GPIOA

在 VC5509 DSP 中 EMIF、EHPI 和 GPIOA 是复用 A、C 组管脚的。它们的转换受复位时 GPIO0 管脚的状态与外部总线选择器 EBSR（External Bus Selection Register）的 Parallel Port Mode 位来决定的。其转换关系如下：

□ 在复位时：

当 GPIO0 为高时，A 为 EMIF 的地址输出，C 为 EMIF 的控制总线；

当 GPIO0 为低时，A 为 HPI 的地址输出，C 为 HIP 的控制总线输出；

本开发板通过跳线 J2 对 GPIO0 引脚进行上拉和下拉，以实现不同的配置。

□ 在复位后：其选择是由 EBSR 的 Parallel Port Mode 位决定

当 Parallel Port Mode = 00 时：

数据 EMIF 方式，D 为数据总线，C 为 EMIF 的控制总线；A 为 GPIO；

当 Parallel Port Mode = 01 时：

全 EMIF 方式，D 为数据总线，C 为 EMIF 的控制总线；A 为 EMIF 的地址输出；

当 Parallel Port Mode = 10 时：

非复用 HPI 方式，D 为数据总线，C 的一部分为 HPI 的控制总线，一部分为 GPIO；A 为 HPI 的地址输出；

当 Parallel Port Mode = 11 时：

复用型 HPI 方式，D 为数据总线，C 的一部分为 HPI 的控制总线，一部分为 GPIO；A 为 GPIO；

外部总线选择寄存器 EBSR(External Bus Selection Register)详细说明如下：

15	14	13	12	11	10	9	8
CLKOUT Disable	OSC Disable	HIDL	$\overline{BKE}$	SR STAT	HOLD	HOLDA	CKE SEL
R/W, 0	R/W, 0	R/W, 0	R/W, 0	R/W, 0	R/W, 0	R/W, 1	R/W, 0

7	6	5	4	3	2	1	0
CKE EN	SR CMD	Serial Port2 Mode		Serial Port1 Mode		Parallel Port Mode	
R/W, 0	R/W, 0	R/W, 00		R/W, 00		R/W, 01 if GPIO0 = 1 11 if GPIO0 = 0	

BITS	DESCRIPTION
15	CLKOUT disable. CLKOUT disable = 0: CLKOUT enabled CLKOUT disable = 1: CLKOUT disabled
14	Oscillator disable. Works with IDLE instruction to put the clock generation domain into IDLE mode. OSC disable = 0: Oscillator enabled OSC disable = 1: Oscillator disabled
13	Host mode idle bit. (Applicable **only** if the parallel bus is configured as EHPI.) When the parallel bus is set to EHPI mode, the clock domain is not allowed to go to idle, so a host processor can access the DSP internal memory. The HIDL bit works around this restriction and allows the DSP to idle the clock domain and the EHPI. When the clock domain is in idle, a host processor will not be able to access the DSP memory. HIDL = 0: Host access to DSP enabled. Idling EHPI and clock domain is not allowed. HIDL = 1: Idles the HPI and the clock domain upon execution of the IDLE instruction when the parallel port mode is set to 10 or 11 selecting HPI mode. In addition, bit 4 of the Idle Control Register must be set to 1 prior to the execution of the IDLE instruction.
12	Bus keeper enable.† $\overline{BKE}$ = 0: Bus keeper, pullups/pulldowns enabled $\overline{BKE}$ = 1: Bus keeper, pullups/pulldowns disabled

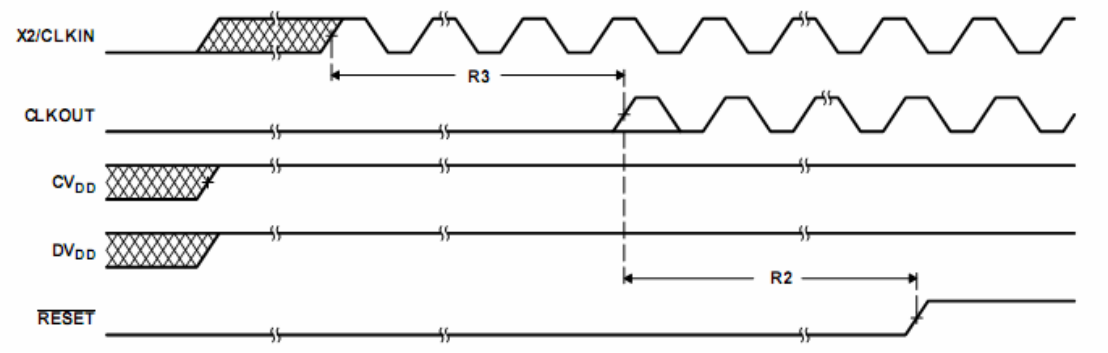
BITS	DESCRIPTION
11	SDRAM self-refresh status bit. SR STAT = 0: SDRAM self-refresh signal is not asserted. SR STAT = 1: SDRAM self-refresh signal is asserted
10	EMIF hold HOLD = 0: DSP drives the external memory bus HOLD = 1: Request the external memory bus to be placed in high-impedance so that another device can drive the memory bus
9	EMIF hold acknowledge. HOLDA = 0: DSP indicates that a hold request on the external memory bus has occured, the EMIF completed any pending external bus activity, and placed the external memory bus signals in high-impedance state (address bus, data bus, CE[3:0], ADE, AWE, ARE, SDRAS, SDCAS, SDWE, SDA10, CLKMEM). Once this bit is cleared, an external device can drive the bus. HOLDA = 1: No hold acknowledge
8	SDRAM CKE pin selection bit. CKE SEL = 0: Use XF for SDRAM CKE signal CKE SEL = 1: Use GPIO.4 for SDRAM CKE signal
7	SDRAM CKE enable bit. CKE EN = 0: XF or GPIO.4 operates in normal mode CKE EN = 1: Based on the CKE SEL bit, either XF or GPIO.4 drives the SDRAM CKE pin
6	SDRAM self-refresh command. SR CMD = 0: EMIF will not issue a SDRAM self-refresh command SR CMD = 1: EMIF will issue a SDRAM self-refresh command
5-4	Serial port2 mode. McBSP2 or MMC/SD2 Mode. Determines the mode of Serial Port2. Serial Port2 Mode = 00: McBSP2 mode. The McBSP2 signals are routed to the six pins of Seral Port2. Serial Port2 Mode = 01: MMC/SD2 mode. The MMC/SD2 signals are routed to the six pins of Seral Port2. Serial Port2 Mode = 10: Reserved Serial Port2 Mode = 11: Reserved.
3-2	Serial port1 mode. McBSP1 or MMC/SD1 Mode. Determines the mode of Serial Port1. Serial Port1 Mode = 00: McBSP1 mode. The McBSP1 signals are routed to the six pins of Seral Port1. Serial Port1 Mode = 01: MMC/SD1 mode. The MMC/SD1 signals are routed to the six pins of Seral Port1. Serial Port1 Mode = 10: Reserved Serial Port1 Mode = 11: Reserved.
1-0	Parallel port mode. EMIF/HPI/GPIO Mode. Determines the mode of the parallel port. Parallel Port Mode = 00: Data EMIF mode. The 16 EMIF data signals and 13 EMIF control signals are routed to the corresponding external parallel bus data and control signals. The 14 (LQFP) or 16 (BGA) address bus signals can be used as general-purpose I/O only. Parallel Port Mode = 01: Full EMIF mode. The 14 (LQFP) or 21 (BGA) address signals, 16 data signals, and 15 control signals are routed to the corresponding external parallel bus address, data, and control signals. Parallel Port Mode = 10: Non-multiplexed HPI mode. The HPI is enabled an its 14 address signals, 16 data signals, and 7 control signals are routed to the corresponding address, data, control signals of the external parallel bus. Moreover, 8 control signals of the external parallel bus are used as general-purpose I/O. Parallel Port Mode = 11: Multiplexed HPI mode. The HPI is enabled and its 16 data signals and 10 control signals are routed to the external parallel bus. In addition, 3 control signals of the external parallel bus are used as general-purpose I/O. The 14 (LQFP) or 16 (BGA) external parallel port address bus signals are used as general-purpose I/O.

注：详细说明参看文档《TMS320VC5509 DSP Host Port Interface(HPI) Reference Guide (SPRU619)》和《TMS320VC5509 Data Sheet (SPRS163)》

2.5 系统复位与中断

2.5.1 系统复位

系统复位为低电平有效，从上电到 DSP 脱离复位，一般需要 30ns+3 个 CLOCKOUT 的输出。设计时复位管脚要加上拉电阻。其时序图如下：



其中 R3 最大为 30ns，R2 为 3 个 CLKOUT。

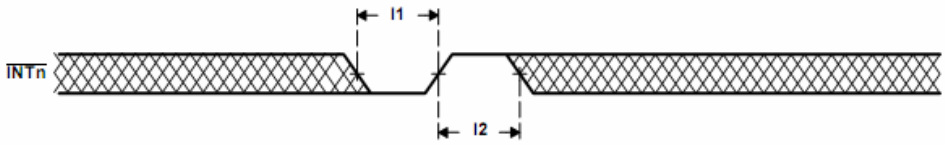
本开发板提供上电复位，无手动复位功能。

2.5.2 中断

VC5509 一共有 5 个外部中断，为 INT[0—4]。中断时序如下：

NO.			CV_{DD} = 1.2 V		CV_{DD} = 1.6 V		UNIT
			MIN	MAX	MIN	MAX	
I1	$t_{w(INTL)A}$	Pulse width, interrupt low, CPU active	3P		3P		ns
I2	$t_{w(INTH)A}$	Pulse width, interrupt high, CPU active	2P		2P		ns

† P = 1/CPU clock frequency in ns. For example, when running parts at 200 MHz, use P = 5 ns.



关于中断的处理方法，VC5509 通过以下的寄存器来管理中断。

Register(s)	Function
IVPD	Points to interrupt vectors 0–15 and 24–31
IVPH	Points to interrupt vectors 16–23
IFR0, IFR1	Indicate which maskable interrupts have been requested
IER0, IER1	Enable or disable maskable interrupts
DBIER0, DBIER1	Configure select maskable interrupts as time-critical interrupts during debugging

其中：

IVPD 与 IVPH 是中断入口的定位寄存器，也就是说 VC5509 的中断向量也是可以重新定位的。

IFR0、IFR1 是中断状态寄存器。

IER0、IER1 是中断使能寄存器。

DBIER0、DBIER1 是用来确定在调试时是否将可屏蔽的中断作为时间敏感的中断处理。

在设置与修改中断向量表时应注意：

在修改中断向量表寄存器 IVPD 与 IVPH 之前，设置 INTM 为 1，阻止外来的中断使程序跑飞。

对于不可屏蔽的中断，应有新旧两个中断向量表，来保证在修改期间，不会执行错误的指令使程序跑飞。

中断向量表的安排如下表：

		Vector Address		
Vector(s)	Interrupt(s)	Bits 23–8	Bits 7–3	Bits 2–0
IV0	Reset	IVPD	00000	000
IV1	Nonmaskable hardware interrupt, NMI	IVPD	00001	000
IV2–IV15	Maskable interrupts	IVPD	00010 to 01111	000
IV16–IV23	Maskable interrupts	IVPH	10000 to 10111	000
IV24	Bus error interrupt (maskable), BERRINT	IVPD	11000	000
IV25	Data log interrupt (maskable), DLOGINT	IVPD	11001	000
IV26	Real-time operating system interrupt (maskable), RTOSINT	IVPD	11010	000
IV27–IV31	General-purpose software-only interrupts INT27–INT31	IVPD	11011 to 11111	000

注：详细说明请参看文档《TMS320VC55x DSP CPU Reference Guide (DPRU371)》

2.6 VC5509 Bootloader

VC5509 的 Bootloader 总共有六种方式，如下：

- 通过 EHPI 的 bootloader，在这种方式下，复用性 HPI 和非复用性 HPI 均可。
- 通过 EMIF 外部异步存储器 Bootloader
- 通过 McBSP0 串口 Bootloader，支持 8 位与 16 位方式。
- 通过 McBSP0 串行 EEPROM Bootloader，支持 16 位与 24 位方式。
- 通过 USB Bootloader

■ 无 Bootloader，直接从片外 FLASH 上执行程序。

Bootloader 引导方式的选择是通过 GPIO[0--3]在复位时的状态来完成的。其说明如下表：

GPIO0	GPIO3	GPIO2	GPIO1	BOOT MODE PROCESS
0	0	0	0	Reserved
0	0	0	1	Serial (SPI) EPROM Boot (24-bit address) via McBSP0
0	0	1	0	USB
0	0	1	1	Reserved
0	1	0	0	Reserved
0	1	0	1	HPI – multiplexed mode
0	1	1	0	HPI – nonmultiplexed mode
0	1	1	1	Reserved
1	0	0	0	Execute from 16-bit-wide asynchronous memory (on $\overline{CE1}$ space)
1	0	0	1	Serial (SPI) EPROM Boot (16-bit address) via McBSP0
1	0	1	0	Reserved
1	0	1	1	16-bit asynchronous memory (on $\overline{CE1}$ space)
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Standard serial boot via McBSP0 (16-bit data)
1	1	1	1	Standard serial boot via McBSP0 (8-bit data)

本开发板上 GPIO0 通过跳线 J2 来选择上拉或下拉，GPIO1 通过跳线 J3 来选择上拉或下拉，GPIO2 通过跳线 J4 来选择上拉或下拉，GPIO3 通过跳线 J5 来选择上拉或下拉。所以，本开发板的 Bootloader 方式可以通过 J2、J3、J4 和 J5 来选择。

注：详细说明请参看文档《TMS320VC5509/C5509A Bootloader (SPRA375)》。

2.7 VC5509 的片上外设

2.7.1 定时器

VC5509 有 2 个 20 位的通用定时器和 1 个看门狗定时器。

定时器的详细说明请参考文档《TMS320VC5509/5510 DSP Timers Reference Guide (SPRU595)》。

2.7.2 DMA

VC5509 共有六个可编程的 DMA 通路。

DMA 的详细说明请参考文档《TMS320VC5509 Direct Memory ACCESS (DMA) Controller Reference Guide (SPRU587A)》。

2.7.3 USB1.1 接口

VC5509 片上有一个符合 USB1.1 标准的接口，但至支持 FULL-SPEED 从方式一种模式，支持的传输类型包括：

- 块传输
- 中断传输
- 同步流传输

本开发板已将该接口连接到一个标准的 USB B 型口上。

USB 的详细说明请参考文档《TMS320VC5509 DSP Universal Serial Bus(USB) Module Reference Guide (SPRU596)》。

2.7.4 IIC 总线

VC5509 上有一个主从两种模式均可的 IIC，本开发板将 IIC 总线作为 CODEC 的控制接口

IIC 的详细说明请参考文档《TMS320VC55x DSP IIC Module Reference Guide (SPRU146)》。

2.7.5 实时时钟 RTC

VC5509 上内含一个实时时钟 RTC 模块，提供年、月、日、时、分、秒等实时时钟信息。本开发板为 VC5509 的 RTC 模块 32.768KHZ 晶体作为时基用，还采用大电容 C5 作为 RTC 的后备电能。在本开发板掉电后，继续为 RTC 提供电源，以保证 RTC 实时时钟的正确。

RTC 的详细说明请参考文档《TMS320VC5509 DSP Real-Time Clock(RTC) Reference Guide (SPRU594)》。

2.7.6 2 通路、10 位 AD

PGE 封装的 VC5509 上有 2 个通道（GHH 封装有 4 个通道）、10 位分辨率的 AD 模拟输入。本开发板才有 PGE 封装的 VC5509，并将 AD 的高基准电压接+3.3V，低基准电压接地，所以 2 通道模拟输入量程为 0 – 3.3v，2 通道模拟输入引至总线扩展连接器 J12 上，供用户自行选用。

AD 的详细说明请参考文档《TMS320VC5509 DSP Analog-to-Digital Converter(ADC) Reference Guide (SPRU586)》。

2.7.7 McBSP 和 MMC/SD 接口

VC5509 片内有 3 个 McBSP 接口和 2 个 MMC/SD 接口，3 个 McBSP 分别为 McBSP0，McBSP1 和 McBSP2，2 个 MMC/SD 接口为 MMC/SD1 和 MMC/SD2.其中 McBSP1 与 MMC/SD1 复用外部引脚，McBSP2 与 MMC/SD2 复用外部引脚。McBSP0 单独使用外部引脚。

在本开发板中，McBSP0 与外扩的 CODEC 器件的数据口接口，实现 CODEC 数据的输入/输出。McBSP2/ MMC/SD2 配置做 MMC/SD 接口，引至 MMC/SD 连接器上，以访问外部 MMC/SD 卡。

有关 McBSP 和 MMC/SD 接口的详细说明请参考文档《TMS320VC55x DSP Multichannel Buffer Serial Port(McBSP) Reference Guide (SPRU592)》和《TMS320VC5509 DSP MultiMediaCard /SD Card Controller Reference Guide (SPRU593)》.

2.7.8 GPIO

VC5509 上用 8 个通用 I/O 的专用引脚 GPIO0 – GPIO7，在 PGE 封装的 VC5509 上没有 GPIO5 引脚。本开发板将 GPIO0 – GPIO4、GPIO6、GPIO7 和 XF 配置为输出，当输出为“0”时，对应的 LED 指示灯就会点亮。输出信号与 LED 指示灯之间的对应关系如下表所示：

GPIO0	GPIO1	GPIO2	GPIO3	GPIO4	GPIO6	GPIO7	XF
D12	D11	D10	D9	D8	D7	D6	D5

第三章 外扩 SDRAM 存储器

本开发板上已通过 VC5509 的 EMIF 总线扩展了一片 SDRAM，在标准配置时，板上安装一片 64M 位（4M*16 位）的 SDRAM，它将占用 VC5509 的 CE0 和 CE1 两个片外存储空间，可寻址范围为 0x040000 – 0x7fffff；最大配置时，板上可安装一片 128M 位（8M*16 位）的 SDRAM，它将占用 VC5509 的 CE0 – CE3 全部四个片外存储空间，可寻址范围为 0x040000 – 0xffffff(MPNMC=1 时)，上电复位时，MPNMC 被清为 0。根据 VC5509 的 EMIF 接口的特点，SDRAM 的工作频率为 CPU 主时钟的一半，主时钟 144MHZ 时，SDRAM 的工作频率为 72MHZ。

在使用 SDRAM 之前，VC5509 还需要对 EMIF 进行配置，其配置过程如下：

- 设置寄存器 EBSR，将外部总线设为 EMIF 工作模式
- 设置寄存器 CEx，选择 SDRAM 容量的大小、数据的宽度、刷新方式等
- 设置有关时序的寄存器。包括 SDC1、SDPER、SDCNT、SDC2。

EMIF 的相关设置，一般采用 CLS 库中所提供的函数比较方便。详细的情况请参看程序 EMIF 的例程

有关 EMIF 接口的详细说明请参考文档《TMS320VC5509 DSP External Memory Interface(EMIF) Reference Guide (SPRU670)》

第四章 音频输入与输出

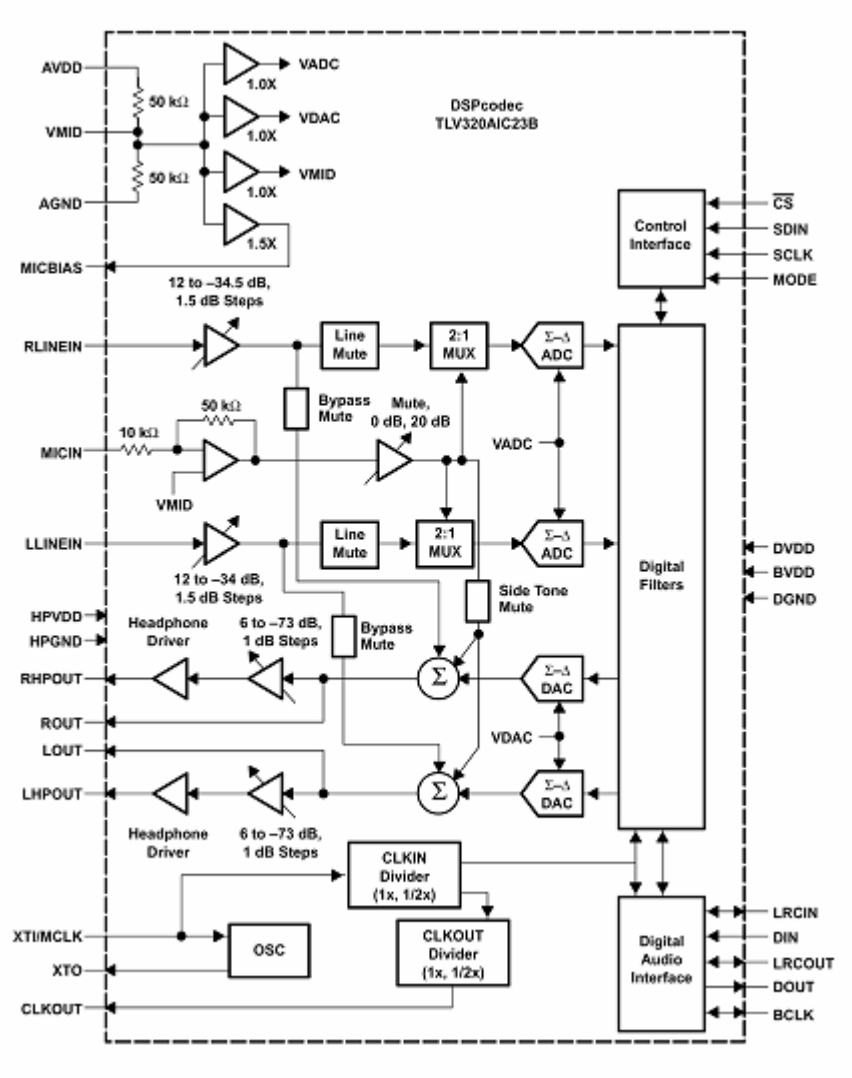
本开发板擦用 TLV320AIC23B 实现音频的输入与输出，TLV320AIC23B 是一片音频 CODEC 器件，它的基本功能包括：48KHZ 带宽、96KHZ 采样率、双声道立体声 AD、DA；音频输出包括：立体声输出（提供耳机功率放大器，能提供 30mW 输出功率，驱动 32 欧姆负载）。

TLV320AIC23B 与微处理器的接口有两个，一个是控制口，用于设置 TLV320AIC23B 的工作参数；另一个是数据口，用于传输 TLV320AIC23B 的 AD、DA 数据。

VC5509 用 IIC 总线接口与 TLV320AIC23B 控制口连接，实现对 TLV320AIC23B 各寄存器的设置。VC5509 还将片上外设 McBSP0 配置为 IIS 工作方式，接口 TLV320AIC23B 的数据口，实现音频数据的输入/输出。

4.1 TLV320AIC23B

TLV320AIC23B（以下简称为 AIC23B）是 TI 推出的一款高性能的立体声音频 CODEC 芯片，内置耳机输出功率放大，支持 MIC 和 LIN IN 两种输入方式（二选一），并且输入和输出都具有可编程增益调节。AIC23B 的模数转换 ADC 和数模转换 DAC 部件高度集成在芯片内部，采用了先进的 Sigma-Delta 过采样技术，可以在 8K 到 96K 采样率范围内提供 16 位、20 位、24 位和 32 位采样，ADC 和 DAC 的信噪比分别可到达 90dB 和 100dB。同时，AIC23B 还具有很低的功耗，回放模式下功率仅为 23mW，省电模式下更是少于 15uW。AIC23B 的管脚和内部结构框图如下：



4.2 TLV320AIC23B 与微处理器的接口

TLV320AIC23B 与微处理器的接口有两个，一个是控制口，用于设置 TLV320AIC23B 的工作参数；另一个是数据口，用于传输 TLV320AIC23B 的 AD、DA 数据。

本开发板用 VC5509 的 IIC 总线接口与 TLV320AIC23B 控制口连接，McBSP0 配置为 IIS 工作方式，连接 TLV320AIC23B 的数据口。

4.2.1 AIC23B 的数据口

TLV320AIC23B 的数据口有四种工作方式，分别为：

- Right justified
- Left justified
- IIS Mode
- DSP Mode

其中后两种可以很方便地与 DSP 的 McBSP 串口相连接。下面我们以 DSP Mode 模式说明数据口的连接。其硬件上的管脚说明如下：

BCLK：数据口位-时钟信号。TLV320AIC23B 为从模式（一般情况），该时钟由 DSP 产生；TLV320AIC23B 为主模式时，该时钟由 TLV320AIC23B 产生

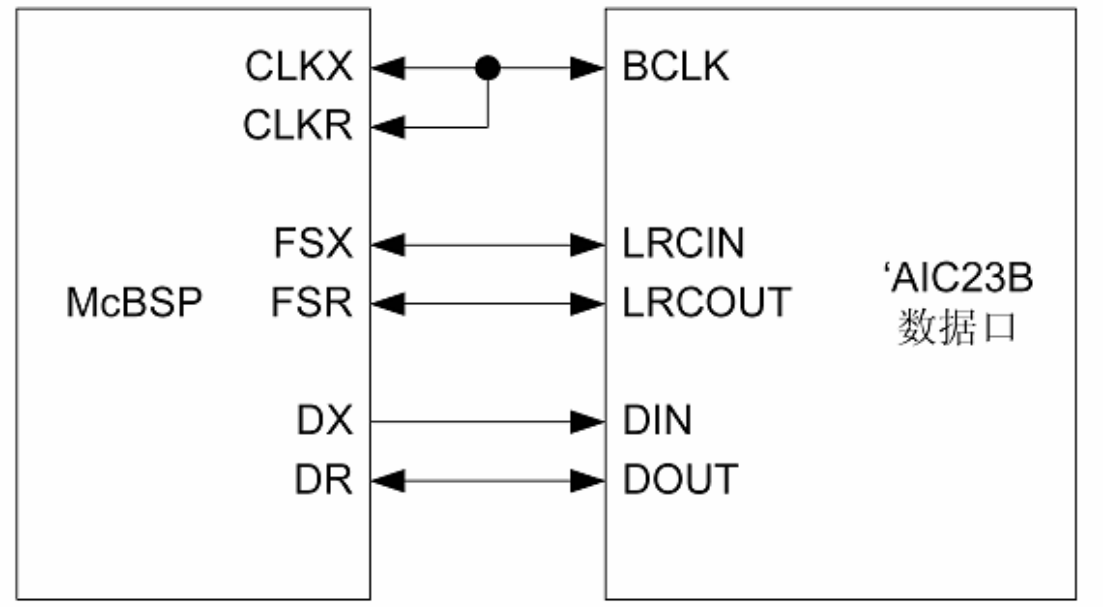
LRCIN：数据口 DAC 输出的帧同步或 IIS 模式下的左/右声道时钟

LRCOUT：数据口 ADC 输入的帧同步或 IIS 模式下的左/右声道时钟

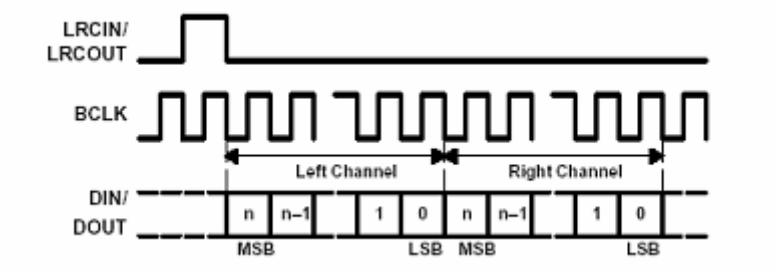
DIN：数据口 DAC 输出的串行数据输入

DOUT：数据口 ADC 输入的串行数据输入

这部分可以和 DSP 的 McBSP 无缝连接，唯一要注意的地方是 McBSP 的接收时钟和 AIC23B 的 BCLK 都由 McBSP 的发送时钟提供；当 AIC23B 做主设备时，McBSP 的发送与接收时钟均由 AIC23B 来提供。连接示意图如下：



DSP 与 AIC23B 的连接既可以采用 DSP 模式，也可以采用 IIS 模式，区别仅在于 DSP 的 McBSP 帧同步信号的宽度。后者的帧同步信号的宽度必须为一个字（16 位）长，而前者的帧宽度可以为一个位长，比如在字长 16 位（即左右声道的采用各为 16 位），帧长为 32 位的情况下，如果采用 IIS，帧同步信号宽度应为 16 位，而采用 DSP Mode 帧信号宽度 1 位即可。DSP 模式时，AIC23B 数据口的时序如下所示：



McBSP 的详细说明请参考文档《TMS320VC55x DSP Multichannel Buffer Serial Port(McBSP) Reference Guide (SPRU592)》

4.2.2 AIC23B 的控制口

TLV320AIC23B 的控制方式有两种工作方式，分别为：

2 线制的 IIC 方式

3 线制的 SPI 防止

VC5509 片上含有 IIC 模块，所以本开发板直接采用 VC5509 的 IIC 总线来连接 TLV320AIC23B 的控制口。VC5509 配置为 IIC 的主设备，TLV320AIC23B 位 IIC 的从设备，VC5509 通过 IIC 总线，对 TLV320AIC23B 进行配置。

TLV320AIC23B 中可配置的控制寄存器如下表所示：

地址	需要初始化的寄存器
0000000	左声道的音量控制寄存器
0000001	右声道的音量控制寄存器
0000010	左声道耳机的音量控制寄存器
0000011	右声道耳机的音量控制寄存器
0000100	模拟音频的路径控制寄存器
0000101	数字音频的路径控制寄存器
0000110	省电方式控制寄存器
0000111	数字音频的接口格式寄存器
0001000	采样率的设置寄存器
0001001	数字接口设置寄存器
0001111	复位寄存器

4.3 TLV320AIC23B 的模拟接口

TLV320AIC23B 的模拟接口主要包括以下四个部分：

■ 立体声输入

■ MIC 输入

■ 立体声输出

■ 耳机输出

4.3.1 音频输入

4.3.1.1 立体声输入

立体声输入口包括左右声道的输入，其管脚为：

LLINEIN：左声道 LINE IN 输入；

RLINEIN：右声道 LINE IN 输入；

详细连接方法请参看原理图。

MIC 输入

MIC 输入主要是用来通过无源的麦克风进行现场声音的采集。其中由于麦克风是无源元器件，所以要为其提供偏置电源。其主要管脚如下：

MICBIAS：提供麦克风偏置电压，通常是 3/4 AVDD；

MICIN：麦克风输入，由 AIC23B 结构框图可以看出放大器默认是 5 倍增益。

麦克风详细连接图请参看原理图。

本开发板采用 2 个标准的 3.5mm 音频插座来进行立体声输入（J19）和麦克风输入(J10)

4.3.2 音频输出

4.3.2.1 立体声输出

TLV320AIC23B 音频输出分为 2 路，其中一路为无功率驱动的立体声输出，其输出引脚为

LOUT: 左声道输出

ROUT：右声道输出

4.3.2.2 耳机输出

TLV320AIC23B 内含耳机功率放大驱动电路，可以直接驱动普通的耳机，不需要外部再进行驱动了。其输出管脚为：

LHPOUT： 左声道耳机放大输出

RHPOUT： 右声道耳机放大输出

本开发板采用 2 个标准的 3.5mm 音频插座用作立体声输出和耳机输出。

4.4 主时钟

本开发板采用 12MHZ 时钟信号为 AIC23B 提供主时钟 MCLK，主时钟 MCLK 与采样率之间的关系请参看文档《TLV320AIC23B Data Manual (SLWS106G)》

第五章 测试程序

本开发板的测试程序有 20 个，分别是

- CPU 看门狗实验
- LED 跑马灯实验
- CPU Timer 定时器实验
- 实时时钟实验
- AD 转换实验
- 扩展 SDRAM 读写实验
- 扩展 FLASH 读写实验
- 快速傅立叶变换(FFT)实验
- FIR 滤波器实验
- IIR 滤波器实验
- 自适应滤波器（FIRLMS）实验
- 键盘扫描实验
- 外部中断输入实验
- AIC23 播音实验
- LCD 显示实验
- 串口通信实验
- USB2.0 通信实验
- 网络通信实验
- MMC/SD 卡读写实验

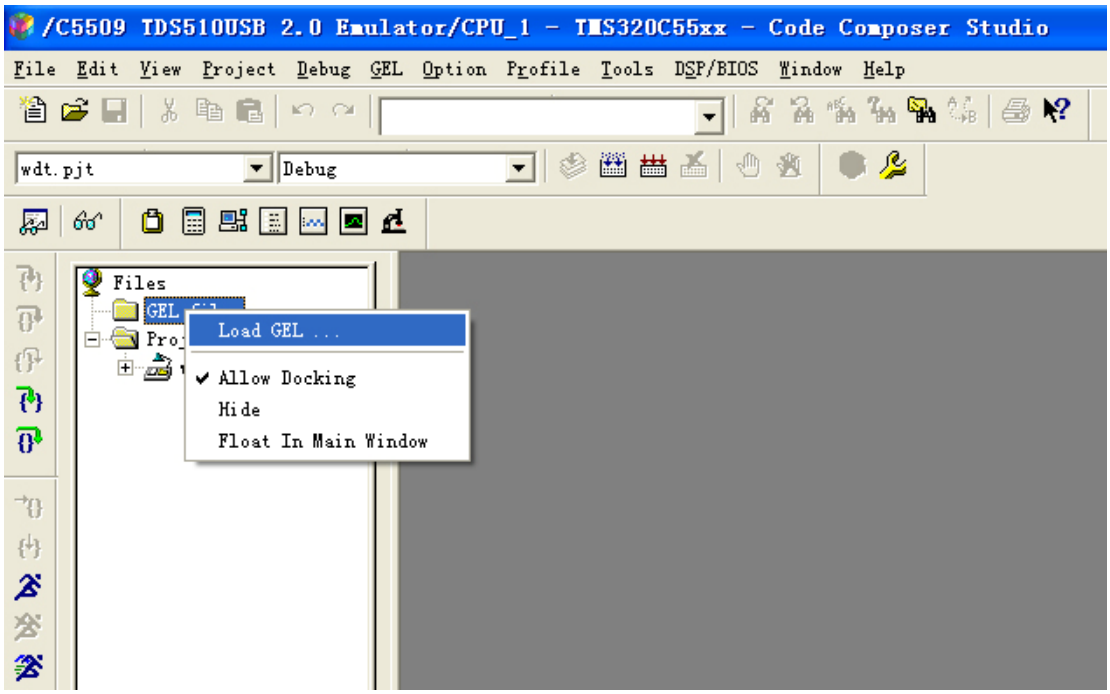
以上测试程序都可以在资料光盘中找到相应的文件夹。测试师，将相应的文件夹拷贝到硬盘上（路径最好不要有空格、汉字或其他特殊字符），并去除只读属性。

测试程序的工作调试环境是基于本公司 TDS510 仿真器的。如果使用其他类型的仿真器，请参看该仿真器的说明文档。

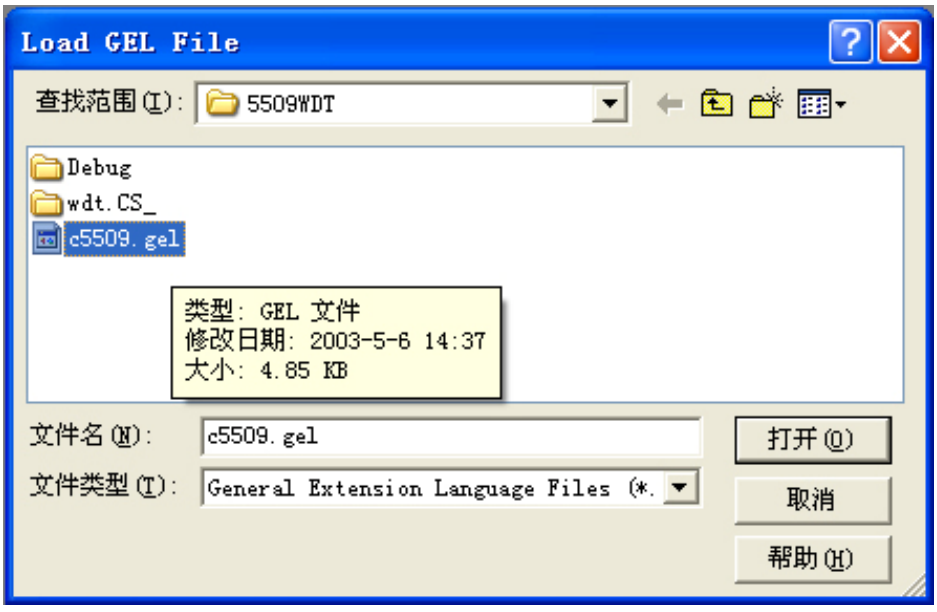
所有程序加载 OUT 文件前，请先加载该文件夹下的 C5509.GEL 文件，以初始化 5509。方法如下：

点击“CCS”，启动 Code Composer Studio 开发环境

点击【project】--【Open…】打开某个工程后，在 CCS 的左边白色 VIEW 框中右击【GEL files】。如下图所示：



选择下拉菜单中【Load GEL…】，在弹出的对话框中选择该工程目录下的“C5509.GEL”，如下图所示，点击【打开】。



实验一、CPU 看门狗实验

这个例子是用来控制看门狗定时器工作的例程。

使用时，按以下步骤进行：

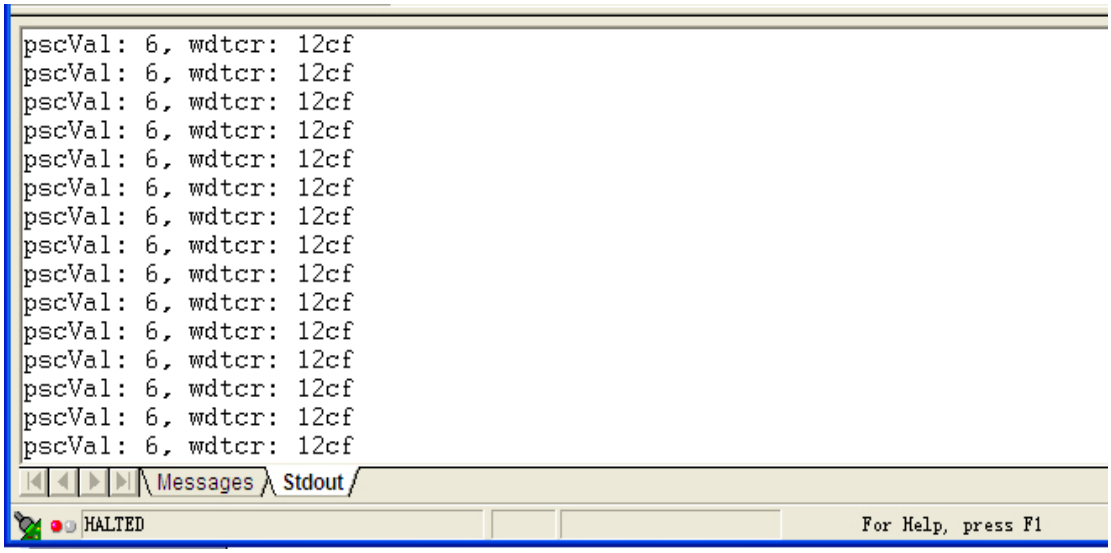
（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509WDT”文件夹下的“WDT.PJT”。

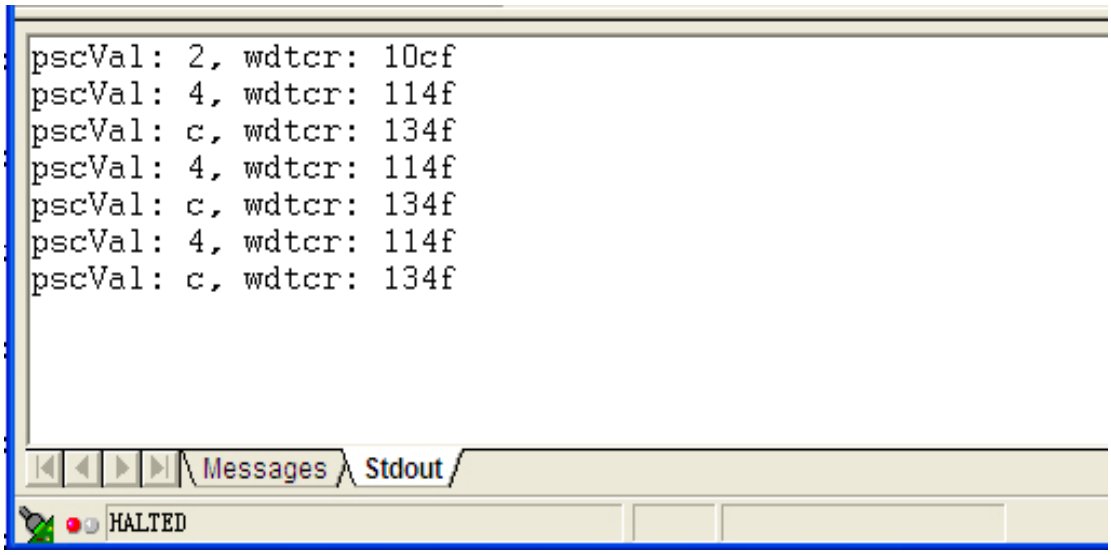
（3）加载“5509WDT”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→ 【Load Program…】，选择“5509WDT”文件夹下的“DEBUG”中的“WDT.OUT”文件，点击【打开】。

（5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。即可看到 CCS 下方“STDOUT”框中，出现看门狗定时器中断信息。如下图所示。



若加载.OUT 文件后，点击【Debug】→【Go Main】，然后点击键盘上的【F10】或点击【Debug】→【Step Over】以单步调试的方法调试程序运行。即可看到下图所示的信息。



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验二　LED 跑马灯实验

这个例子是用来控制 GPIO，进而控制 LED 工作的例程。

使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509GPIO”文件夹下的“5509GPIO.PJT”。

（3）加载“5509GPIO”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→ 【Load Program…】，选择“5509GPIO”文件夹下的“DEBUG”中的“5509GPIO.OUT”文件，点击【打开】。

（5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。观察开发板，即可看到开发板上的 8 个 LED 轮流闪亮。

注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验三 CPU Timer 定时器实验

这个例子是如何使用 VC5509 定时器的例程。

使用时，按以下步骤进行：

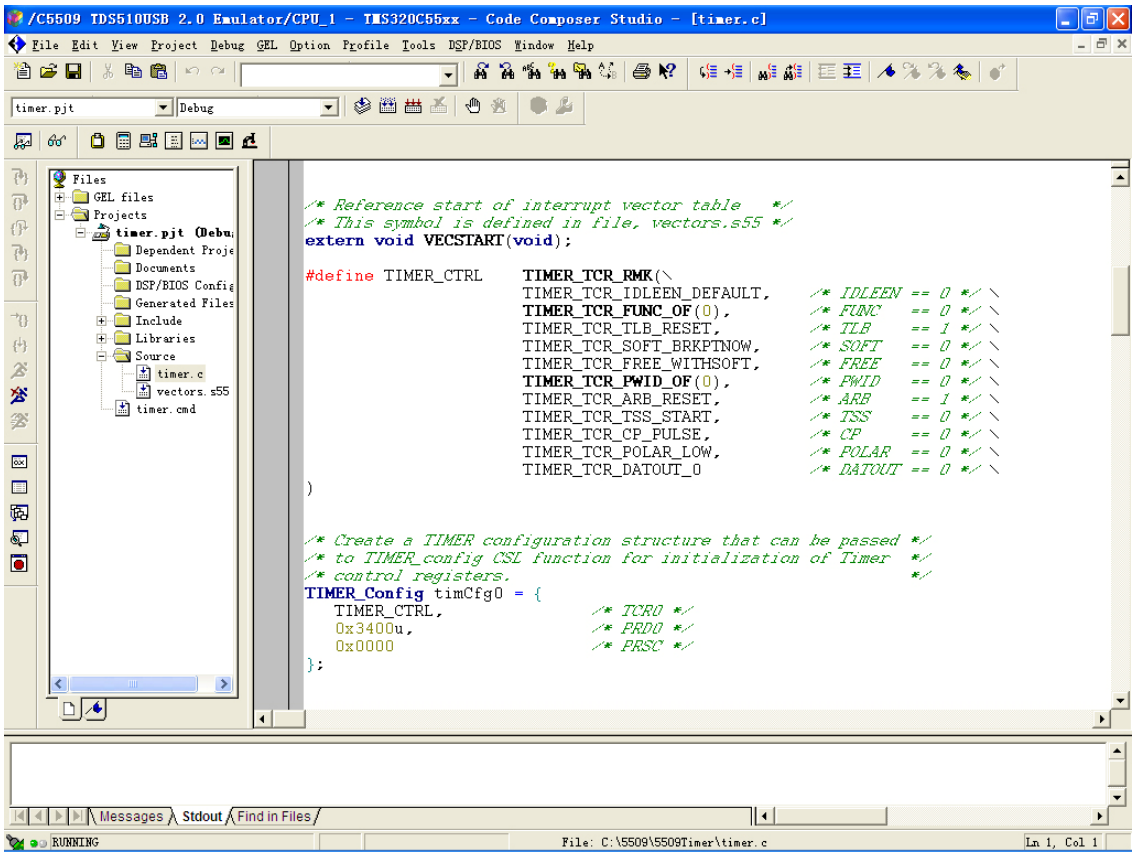
（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509TIMER”文件夹下的“TIMER.PJT”。

（3）加载“5509TIMER”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→ 【Load Program…】，选择“5509TIMER”文件夹下的“DEBUG”中的“TIMER.OUT”文件，点击【打开】。

（5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。观察开发板，即可看到开发板上的D3 快速闪动，表示定时器运作正常。



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验四　　实时时钟实验

这个例子是如何使用 VC5509 RTC 的例程。

使用时，按以下步骤进行：

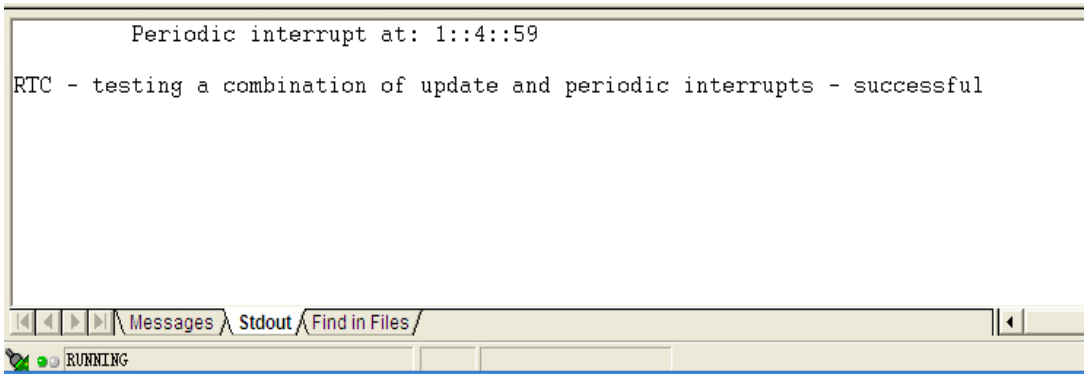
（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509RTC”文件夹下的“RTC.PJT”。

（3）加载“5509RTC”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→ 【Load Program…】，选择“5509RTC”文件夹下的“DEBUG”中的“RTC.OUT”文件，点击【打开】。

（5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。等待一分钟后，RTC 中断产生，即可在 CCS 下方即可看到如下的输出信息，表示 RTC 运作正常。



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验五　AD 转换实验

这个例子是如何使用 VC5509 片上 AD 的例程。

使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

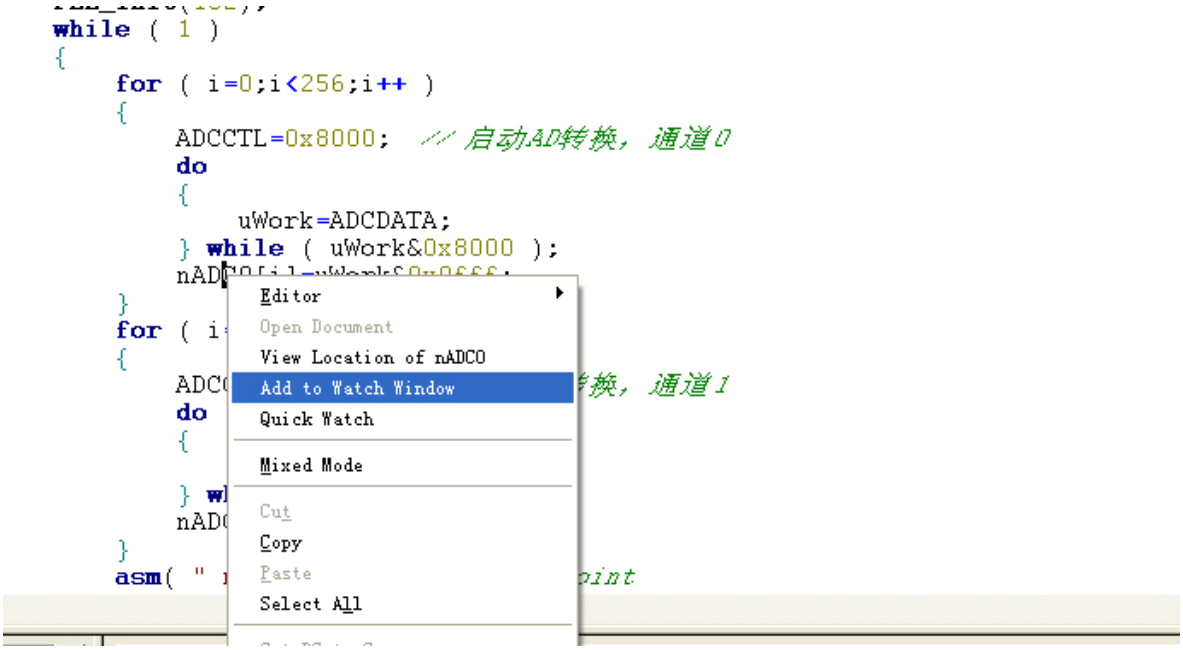
（2）点击【project】→【Open…】打开“AD”文件夹下的“AD.PJT”。

（3）加载“AD”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→【Load Program…】，选择“AD”文件夹下的“DEBUG”中的“AD.OUT”文件，点击【打开】。

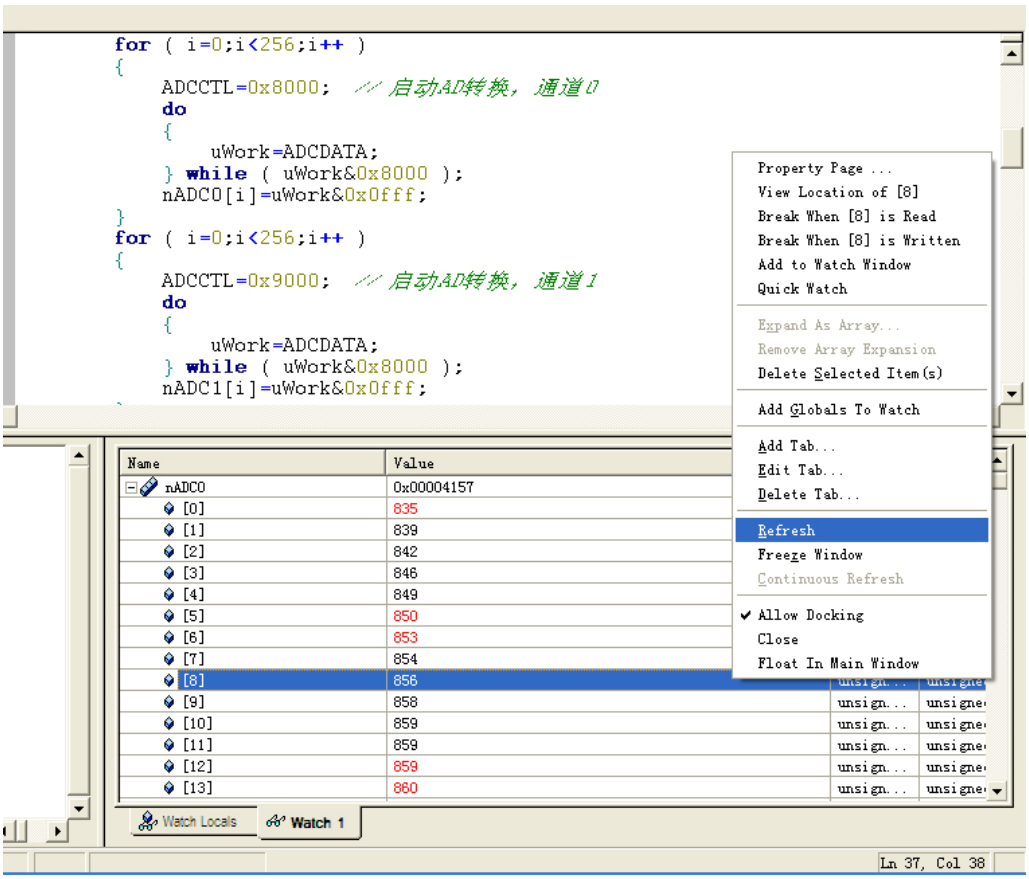
（5）在“break point”处设置断点

（6）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。在“MAIN.C”文件中，找到“nADC0”或“nADC1”，点击鼠标右键，在弹出的下拉菜单中选择【Add to Watch Window】，如下图所示。



这时，CCS 右下方即出现一个“Watch”窗口，在该窗口中即可观察 nADC0 或 nADC1 数组的数字，即 AD 转换的数字，在该窗口中点击鼠标右键，在弹出的菜单中选择【Refresh】，即可实时刷新数据，看到当前 AD 转换的最新结果。如下图所示。

在本实验中，用户可自行在板子的 J12 扩展口的 7 脚、8 脚外接 0 – 3.3v 的输入电压（切记不能超过 3.3v），并观察 CCS 窗口中 AD 转换数值的变化。



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验六　　扩展 SDRAM 读写实验

这个例子是如何使用 VC5509 外扩 SDRAM 的例程。

使用时，按以下步骤进行：

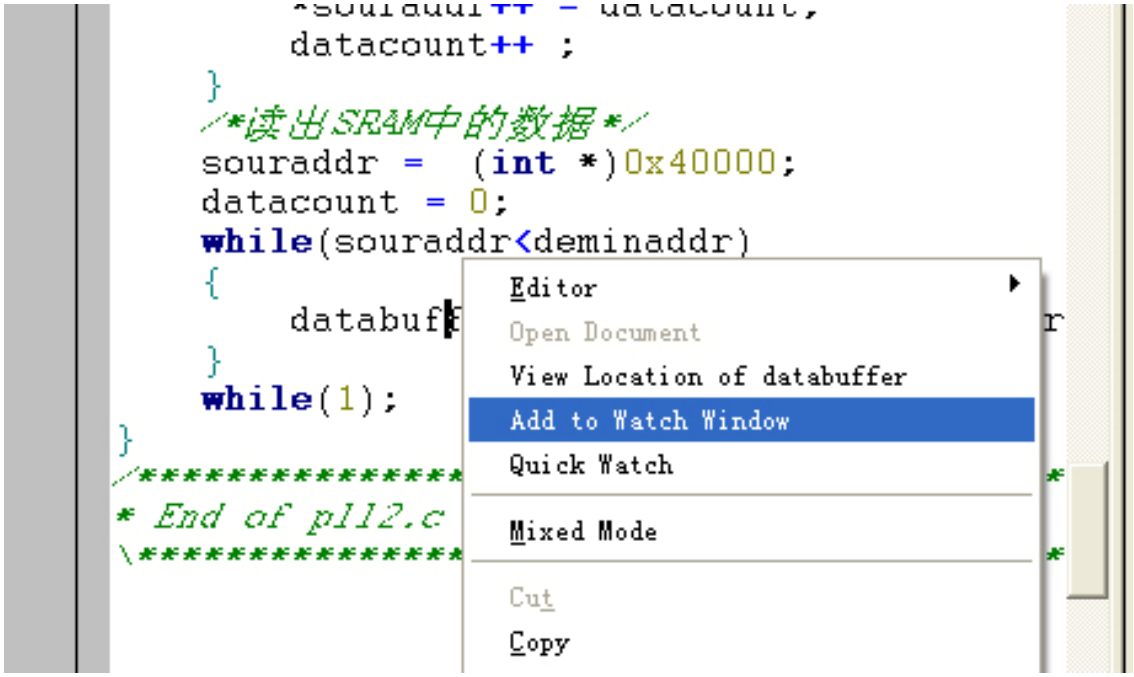
（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509EMIF”文件夹下的“EMIF.PJT”。

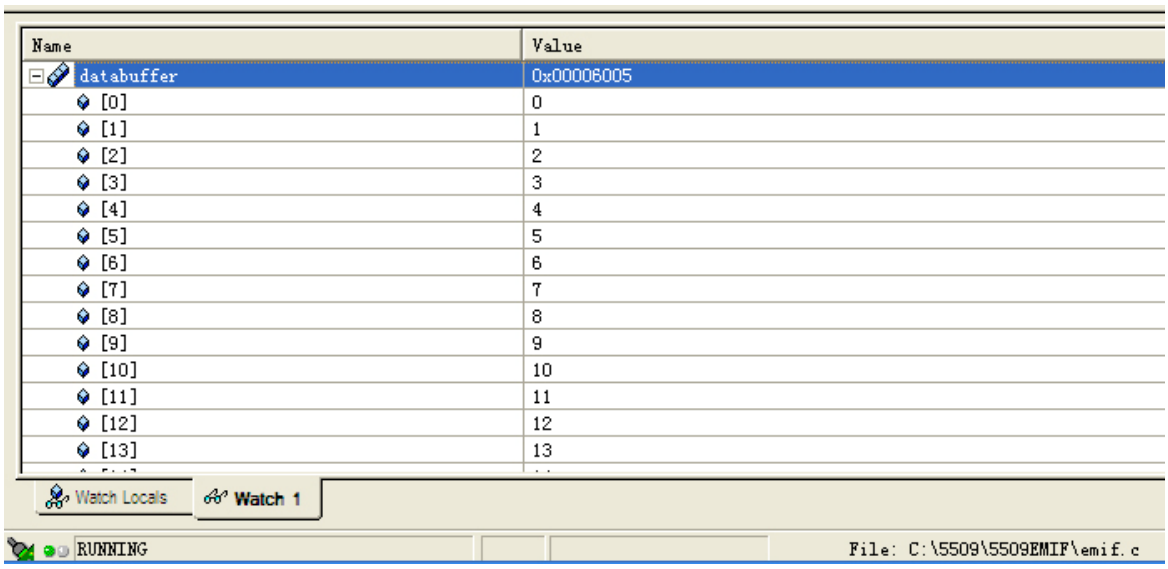
（3）加载“AD”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→ 【Load Program…】，选择“5509EMIF”文件夹下的“DEBUG”中的“EMIF.OUT”文件，点击【打开】。

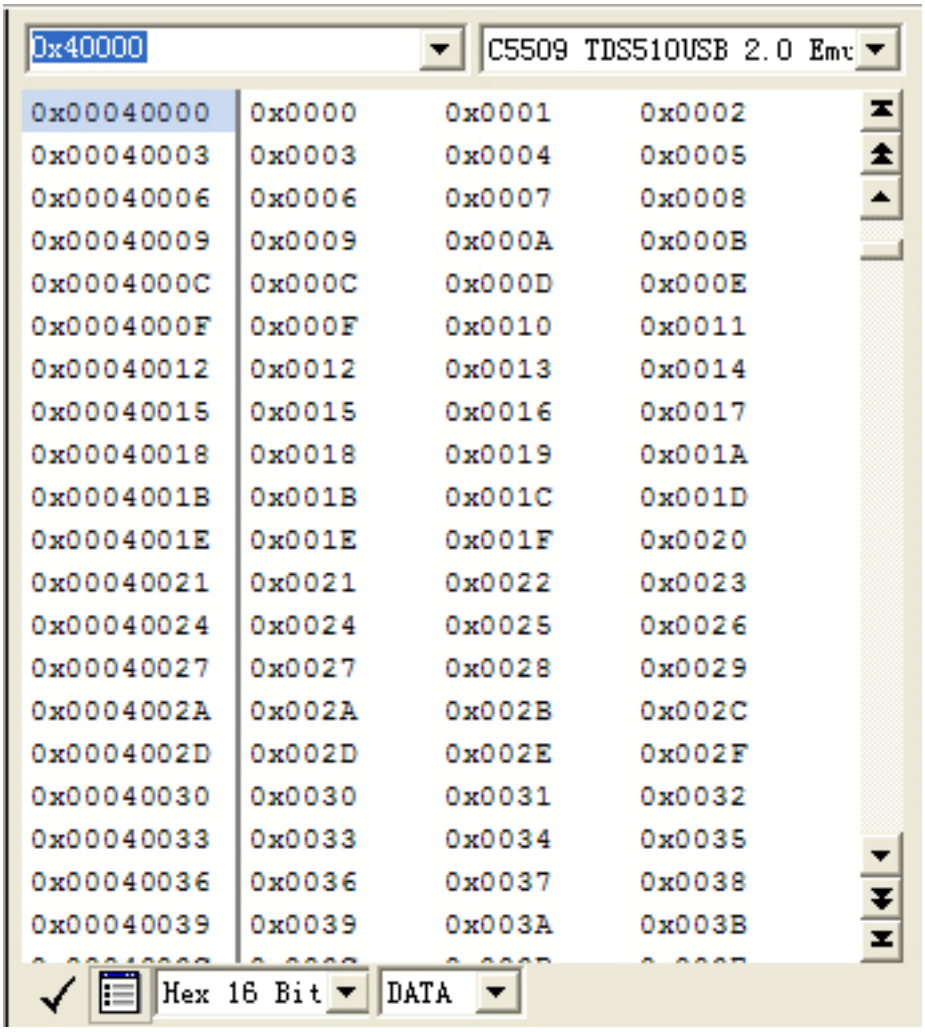
（5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。在“MAIN.C”文件中，找到“databuffer”，点击鼠标右键，在弹出的下拉菜单中选择【Add to Watch Window】，如下图所示。



此时，CCS 下方就会出现一个观察窗，我们可以看到从 SDRAM 读出来的数据，如下图所示，从 1 逐一增大，符合写入的内容，可见 SDRAM 工作正常。



另外，还可以点击【View】→【Memory】或点击 CCS 左侧的快捷图标  ，在打开的内存观测窗口中，其左上方写着“Enter An address”的编辑栏中输入我们刚才写入 SDRAM 的起始地址 0x40000，即可看到 0x40000 等内存地址当中的数据，如下图所示。我们可以看到，数据从 1 逐一增大，符合写入的内容，可见 SDRAM 工作正常。



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验七　扩展 FLASH 读写实验

本实验是如何使用 VC5509 读写片外并行 FLASH 的例程。并行 FLASH 有关的操作时序请参考相应的芯片数据手册。

使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509FLASH”文件夹下的“FLASH.PJT”。

（3）编译程序。

（4）点击【File】→ 【Load Program…】，选择“FLASH”文件夹下的“DEBUG”中的“FLASH.OUT”文件，点击【打开】。

（5）为验证本实验的效果，请按下图所示，打上两个断点。

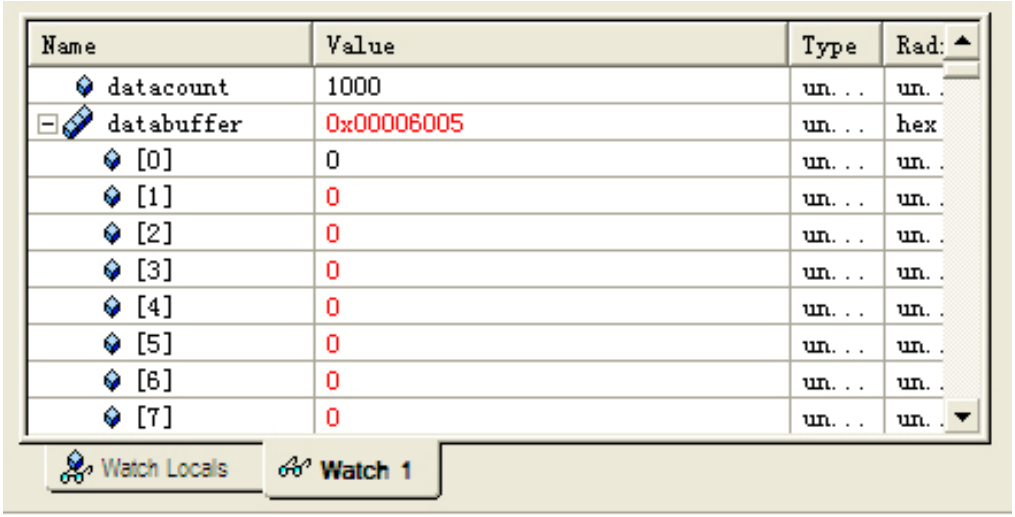
```
    //以下程序为读AM29LV800
    for(datacount=0;datacount<1000;datacount++)
    {
        databuffer[datacount]=0;
    }
    Flash_CS();
    fraddr = fwaddr;
    for(datacount=0;datacount<1000;datacount++)
    {
        databuffer[datacount] = Flash_Read(datacount);
    }
    Flash_disCS();

    //运行到此处，在view--memory里查看databuffer开始地址的数据，
    while(1);
}
```

并在“MAIN.C”文件中，找到“databuffer”，点击鼠标右键，在弹出的下拉菜单中选择【Add to Watch Window】，如此“databuffer”将会添加到左下角的观察框中（本操作请参考上个实验的方法）。

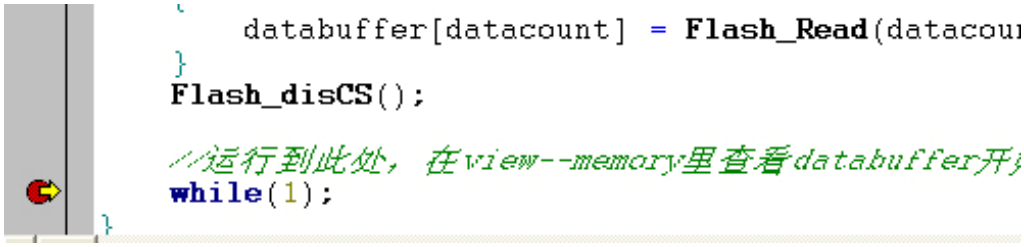
（6）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。当程序运行到第一个断点时，停止在该未知，如下图所示。此时表示：FLASH 写数据的操作已完成，且“databuffer”已全部清零。

```
    //以下程序为读AM29LV800
    for(datacount=0;datacount<1000;datacount+
    {
        databuffer[datacount]=0;
    }
    Flash_CS();
    fraddr = fwaddr;
    for(datacount=0;datacount<1000;datacount+
    {
        databuffer[datacount] = Flash_Read(da
    }
```

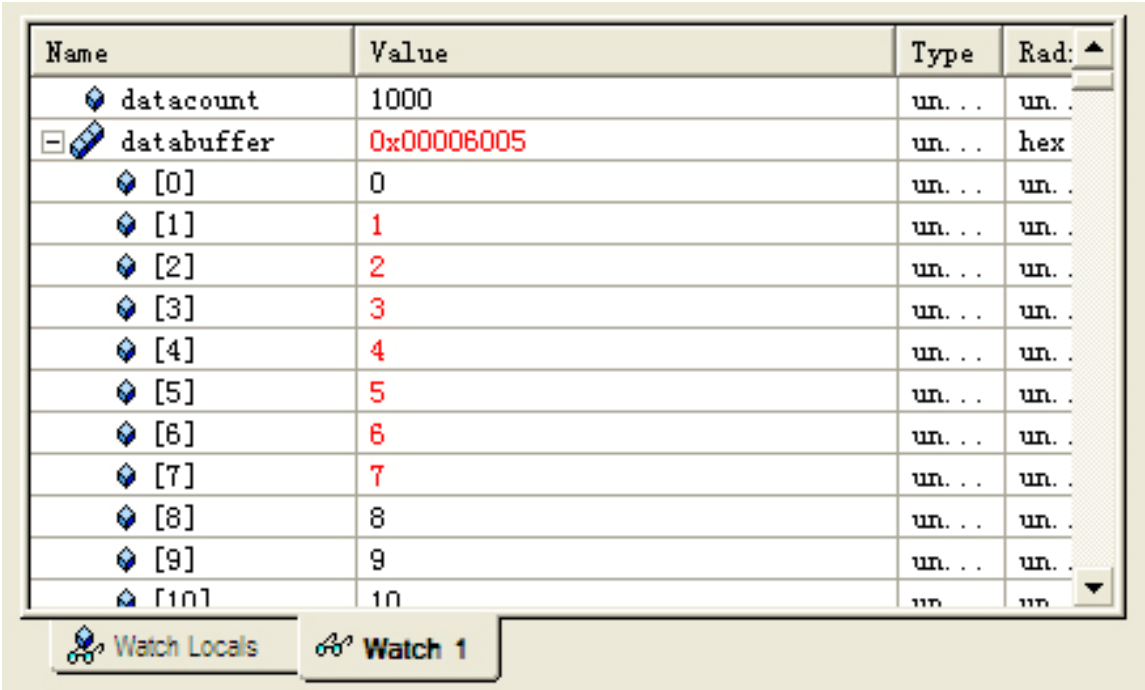
“databuffer”可以在观察框中观察，如下图所示。



此时，再次点击【Debug】→【Run】或左侧快捷键  图标，全速运行。当程序运行到第二个断点时，就会停止此处。如下图所示。此时表示：FLASH 读数据已完成。



查看 FLASH 读出的数据，可在观察框中观察“databuffer”，如下图所示。图中，“databuffer”的数据逐一递增，符合我们程序设计的设想，证明并行 FLASH 的读写是成功的。



注意：本程序中有一段擦除整片 FLASH 的子函数，运行该函数大概要 14S，用户可根据自己的需要自行使用。如下图红圈中所示。

```
/*设置系统的运行速度为144MHz*/
PLL_config(&myConfig);

/*初始化DSP的外部SDRAM*/
EMIF_config(&emiffig);

Flash_Reset();

//Flash_Erase_all运行大约为14s以上，为节约时间注释掉，用户可自行根据需要取消
success = Flash_Erase_all();

//以下程序为烧写AM29LV800
Flash_Write_init();
fwaddr = (int *)CESECT1;  //地址首先指向5509的CE1空间（AM29LV800所在）
fwaddr += 0x10000;        //指向AM29LV800的1扇区
for(datacount=0;datacount<1000;datacount++)
```

注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验八　快速傅立叶变换(FFT)实验

本实验是如何使用 VC5509 进行 FFT 的例程。FFT 有关的理论知识请参考“信号与系统”“数字信号处理”等专业书籍。本实验是纯软件仿真实验。

使用时，按以下步骤进行：

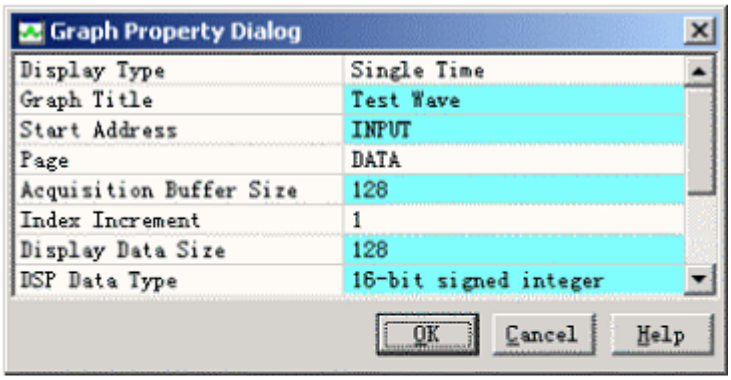
（1）点击“CCS”，启动 Code Composer Studio 开发环境

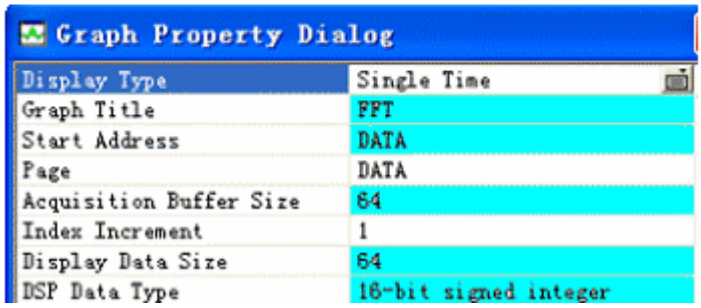
（2）点击【project】→【Open…】打开“FFT”文件夹下的“FFT.PJT”。

（3）编译程序。

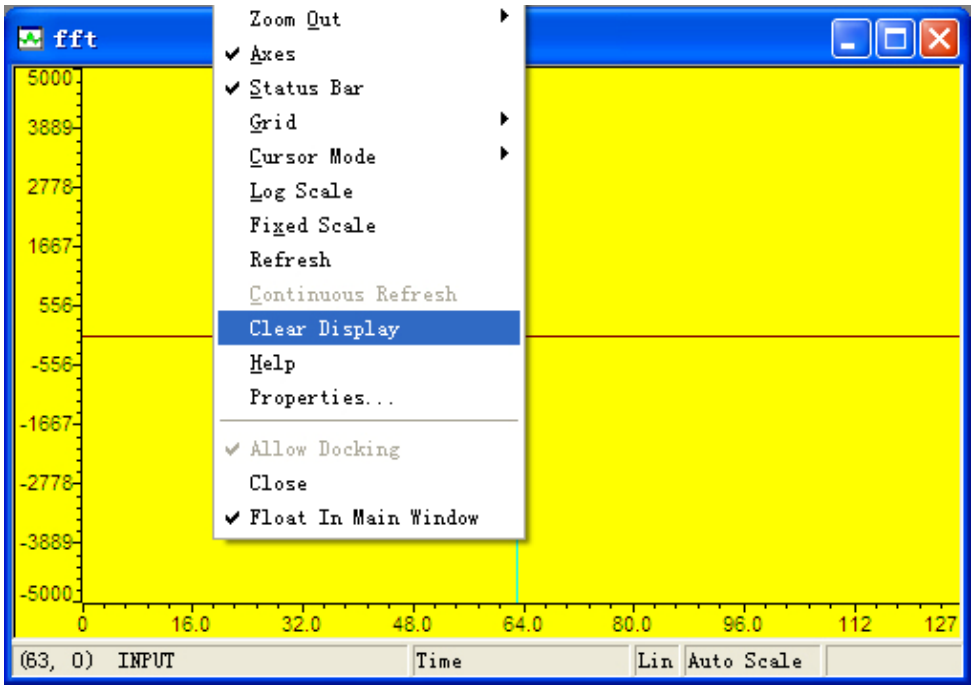
（4）点击【File】→ 【Load Program…】，选择“FFT”文件夹下的“DEBUG”中的“FFT.OUT”文件，点击【打开】。

（5）点击【View】→【Graph】→【Time/Frequency】，打开观察窗口，进行如下图所示的设置，然后点击【OK】。如果出现提示对话框，点击【NO】即可。





（6）清除显示，在如下弹出的图形显示框中，点击鼠标右键，选择弹出菜单中“Clecar Display”。



（7）设置断点：在程序文件“FFT.C”中有注释“break point”的语句上设置软件断点。

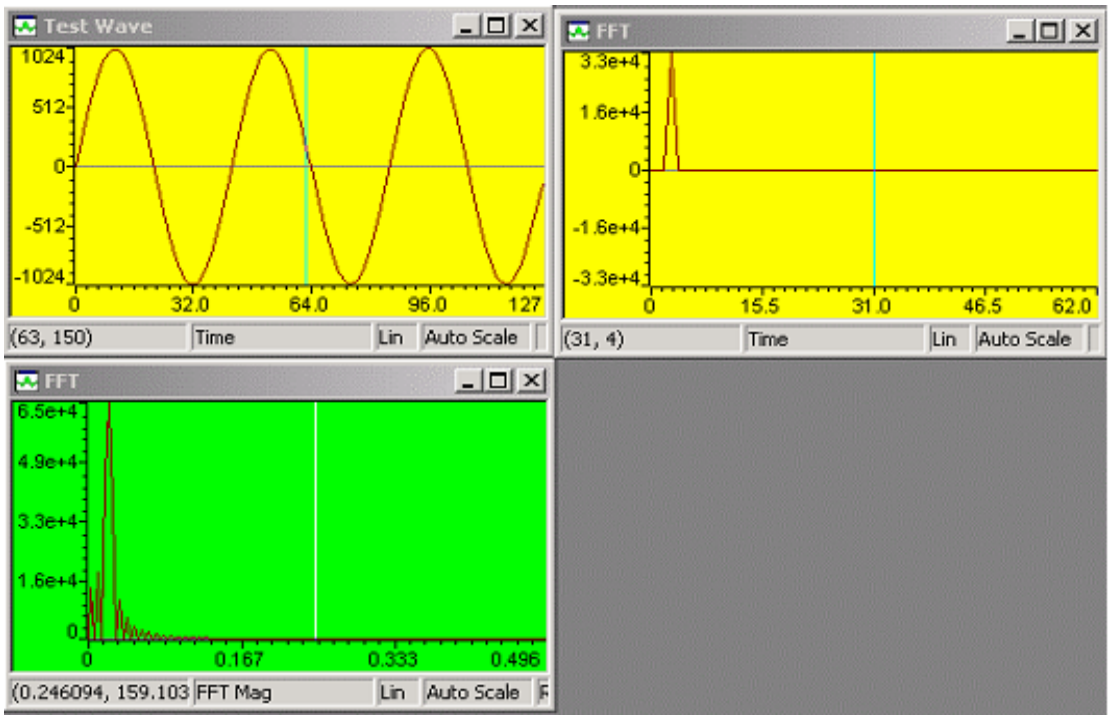
（8）运行并观察结果

A、选择“Debug”菜单中的“Animate”，或按 F12 键运行程序

B、在“FFT”波形窗口中点击右键，选择属性栏 Display Type 更改为 FFT Magnitude 或者是 FFT Magnitude and Phase，更改图形显示为 FFT。观察频域图形。

C 、观察“FFT”波形窗口中的时域图形

D、观察图形窗口中由 CCS 计算出的正弦波的 FFT。如下图所示。



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验九　FIR 滤波器实验

本实验是如何使用 VC5509 进行 FIR 滤波器的例程。FIR 滤波器有关的理论知识请参考“信号与系统”“数字信号处理”等专业书籍。本实验是纯软件仿真实验。

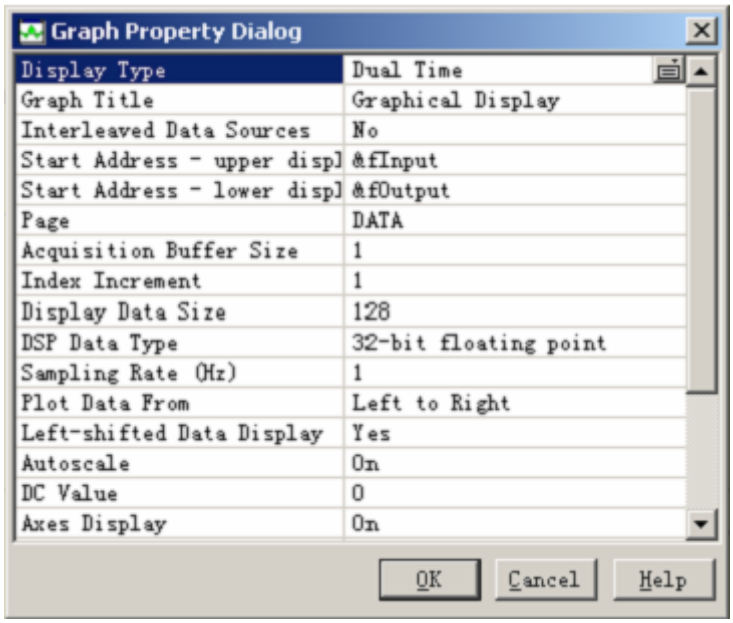
使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“FIR”文件夹下的“FIR.PJT”。

（3）点击【File】→ 【Load Program…】，选择“FIR 文 件夹下的“DEBUG”中的“FIR.OUT”文件，点击【打开】。

（4） 点击【View】→【Graph】→【Time/Frequency】，打开观察窗口，进行如下图所示的设置，然后点击【OK】。



（7）设置断点：在程序文件“FIR.C”中有注释“break point”的语句上设置软件断点。

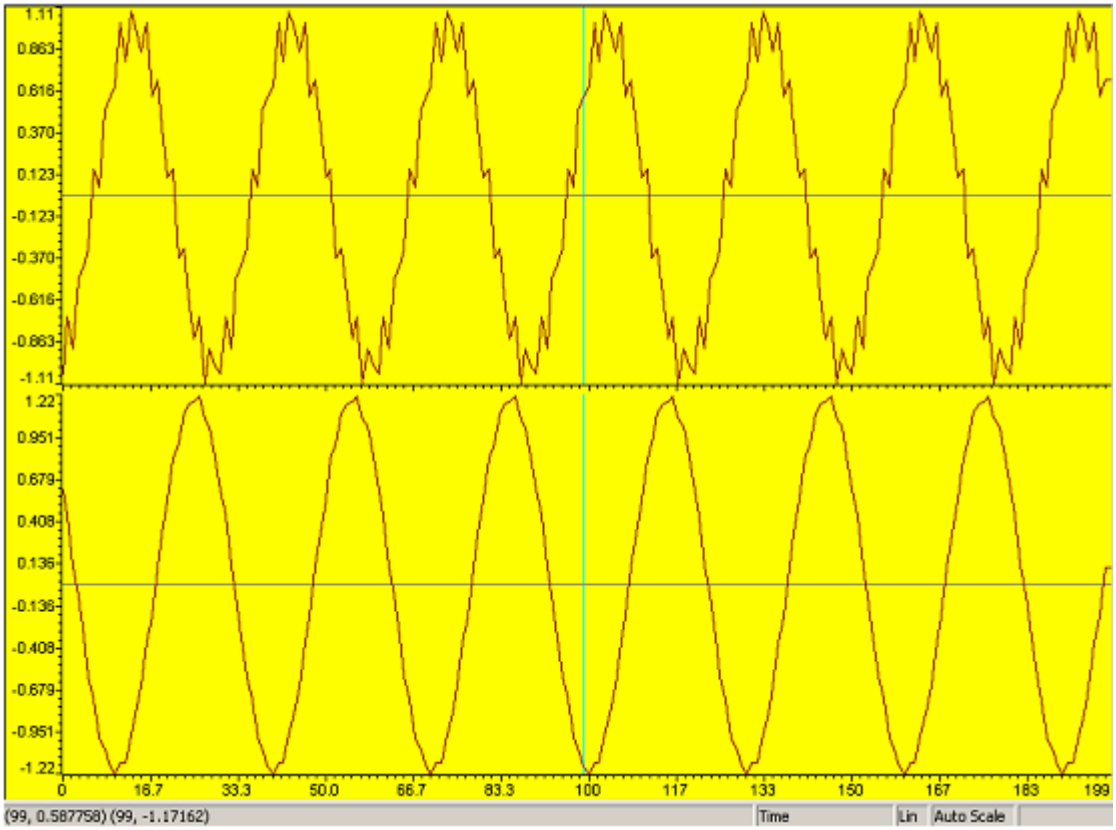
（8）运行并观察结果

A、选择“Debug”菜单中的“Animate”，或按 F12 键运行程序

B、观察“INPUT”“OUTPUT”波形窗口中的时域图形，观察滤波效果。

C、在“INPUT”“OUTPUT”波形窗口中点击右键，选择属性，更改图形显示为 FFT。观察频域图形。

D、观察“INPUT”“OUTPUT”窗口中的频域图形。如下图所示



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十　IIR 滤波器实验

本实验是如何使用 VC5509 进行 IIR 滤波器的例程。IIR 滤波器有关的理论知识请参考“信号与系统”“数字信号处理”等专业书籍。本实验是纯软件仿真实验。

使用时，按以下步骤进行：

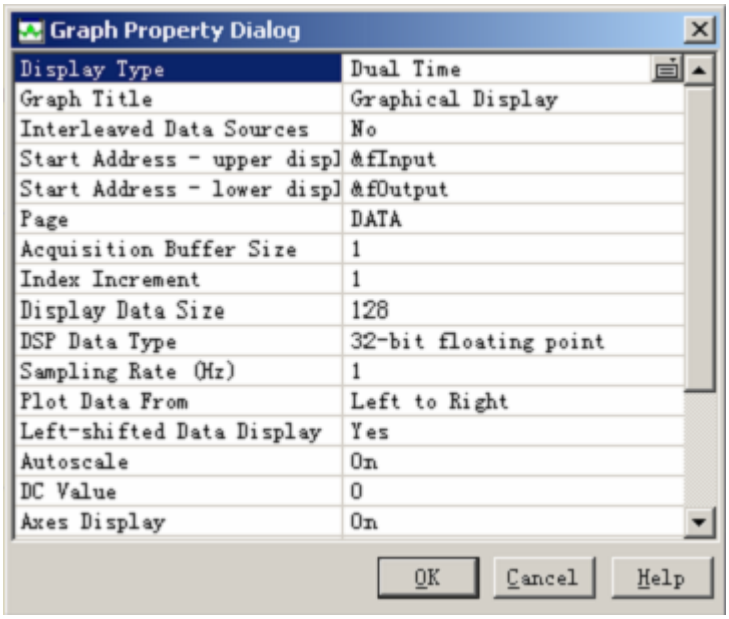
（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“IIR”文件夹下的“IIR.PJT”。

（3）编译程序。

（4）点击【File】→ 【Load Program…】，选择“FIR”文件夹下的“DEBUG”中的“IIR.OUT”文件，点击【打开】。

（5）点击【View】→【Graph】→【Time/Frequency】，打开观察窗口，进行如下图所示的设置，然后点击【OK】。

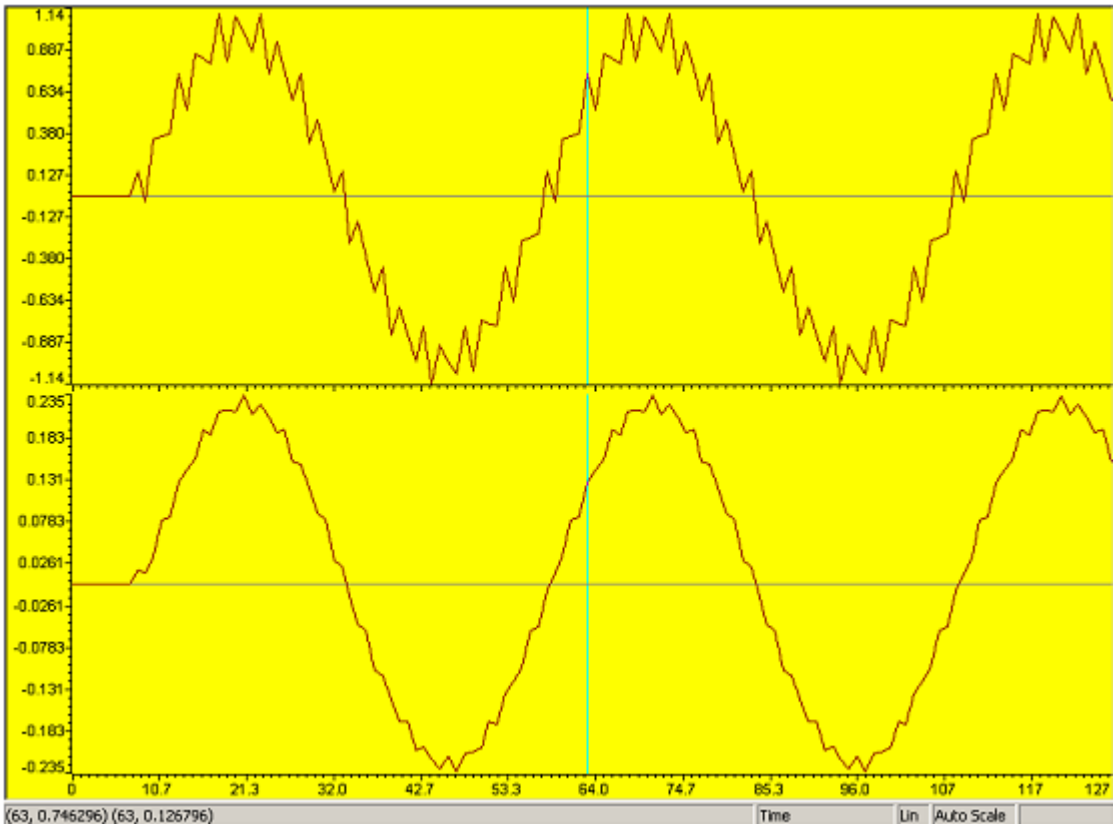


（6）设置断点：在程序文件“IIR.C”中有注释“break point”的语句上设置软件断点。

（7）运行并观察结果

A、选择“Debug”菜单中的“Animate”，或按 F12 键运行程序

B、观察“IIR”波形窗口中的时域图形，观察滤波效果。



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，

然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十一　自适应滤波器（FIRLMS）实验

本实验是如何使用 VC5509 进行 FIRLMS 滤波器的例程。FIRLMS 滤波器有关的理论知识请参考“信号与系统”“数字信号处理”等专业书籍。本实验是纯软件仿真实验。

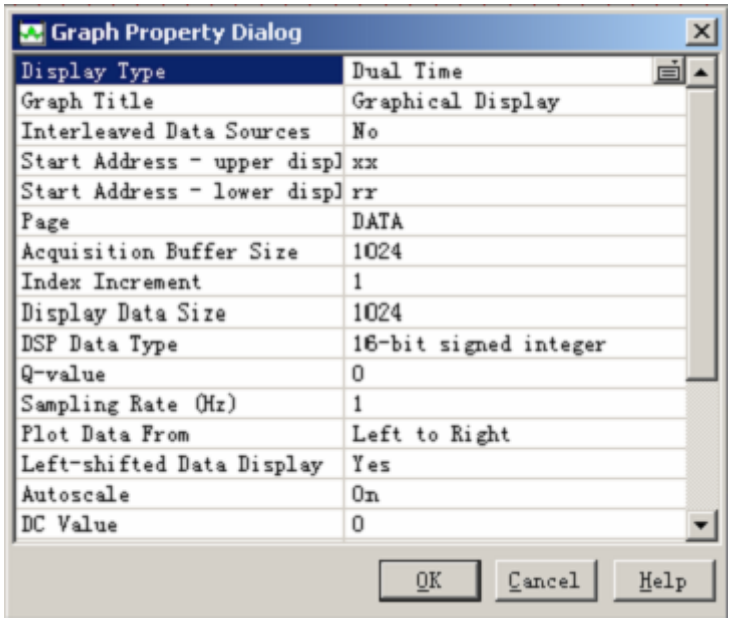
使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“FIRLMS”文件夹下的“FIRLMS.PJT”。

（3）点击【File】→ 【Load Program…】，选择“FIRLMS”文件夹下的“DEBUG”中的“FIRLMS.OUT”文件，点击【打开】。

（4）点击【View】→【Graph】→【Time/Frequency】，打开观察窗口，进行如下图所示的设置，然后点击【OK】。



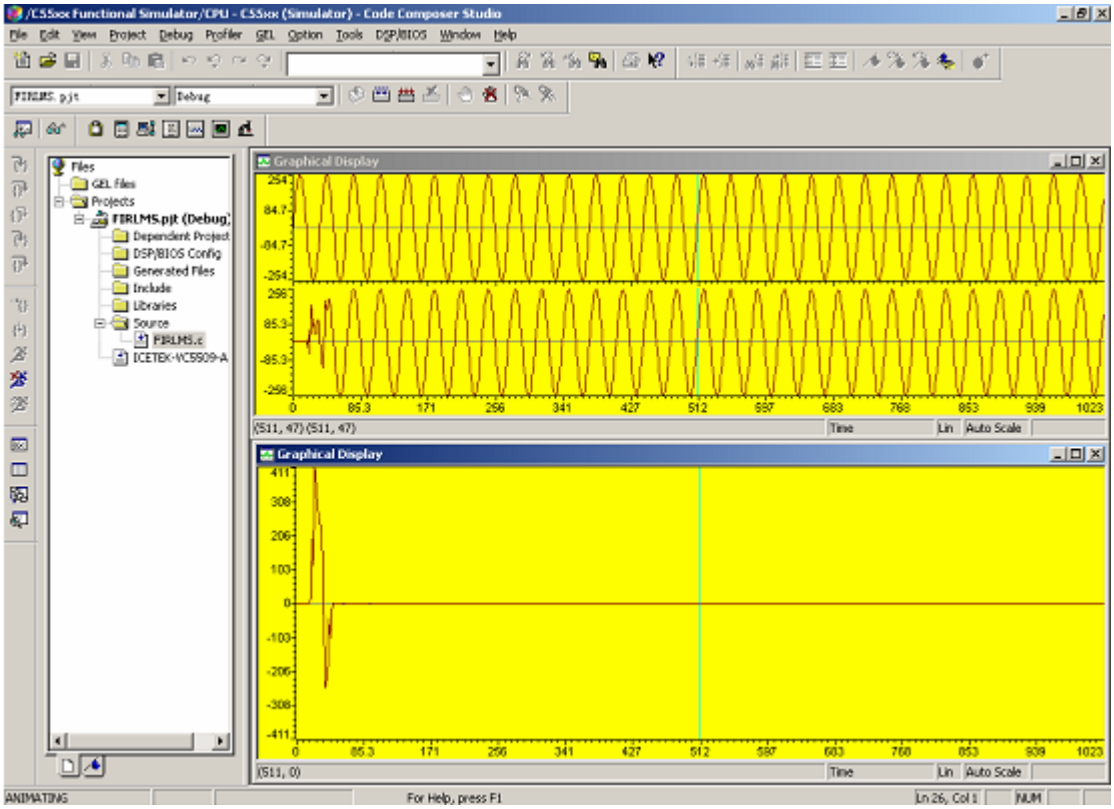
（5）运行并观察结果，在程序的软件断点处加断点。

A、选择“Debug”菜单中的“Animate”，或按 F12 键运行程序

B、当程序运行停止后，观察“IIR”波形窗口中的图形显示结果。

C、选择菜单【File】→【workspace】→【save workspace As】，输入文件名 SY.wks。

实验结果，如下图所示



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十二　键盘扫描实验

本实验是如何使用开发板上 4 个按键的例程。

使用时，按以下步骤进行：

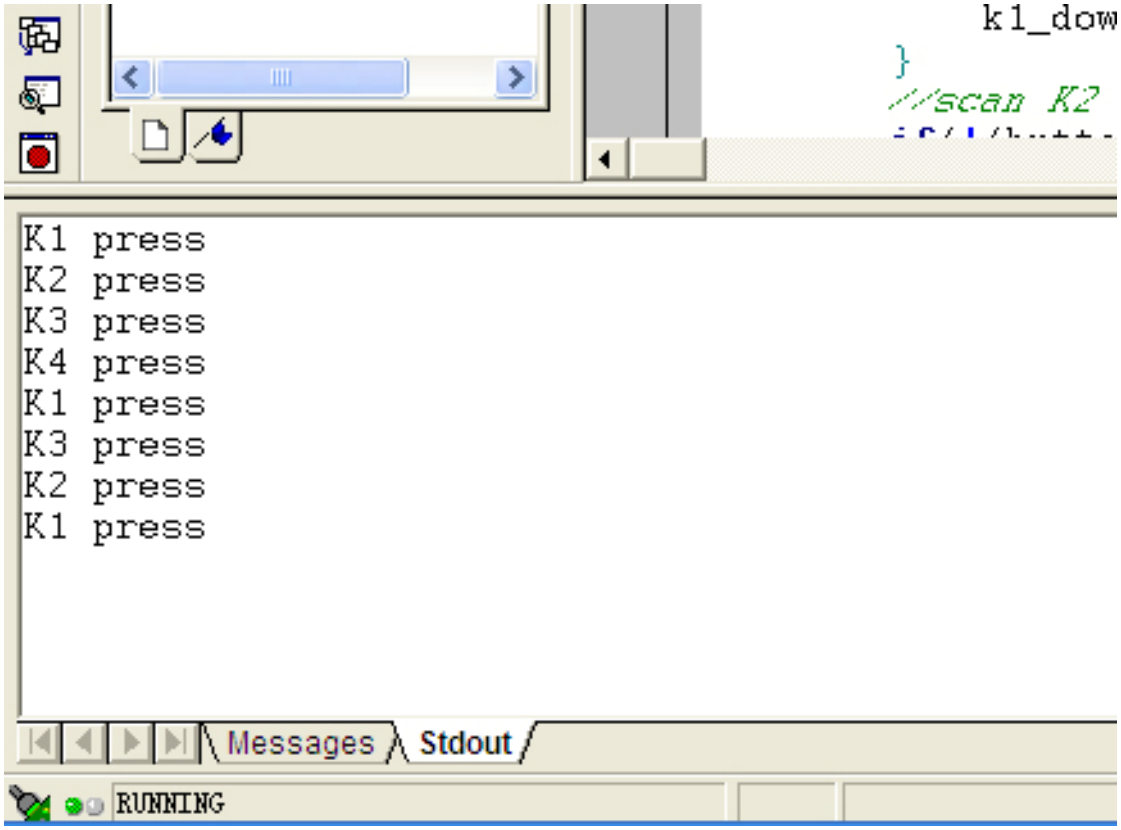
（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509BUTTON”文件夹下的“EMIF_BUTTON.PJT”。

（3）编译程序。

（4）点击【File】→ 【Load Program…】，选择“5509BUTTON”文件夹下的“DEBUG”中的“EMIF_BUTTON.OUT”文件，点击【打开】。

(5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。手按开发板上 K1、K2、K3 或 K4 按键，在 CCS 下方的信息栏中即可看到“K1 presss”之类的提示语，提示某个按键已被按下。如下图所示：



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十三　外部中断输入实验

本实验是如何使用开发板上外部中断的例程。

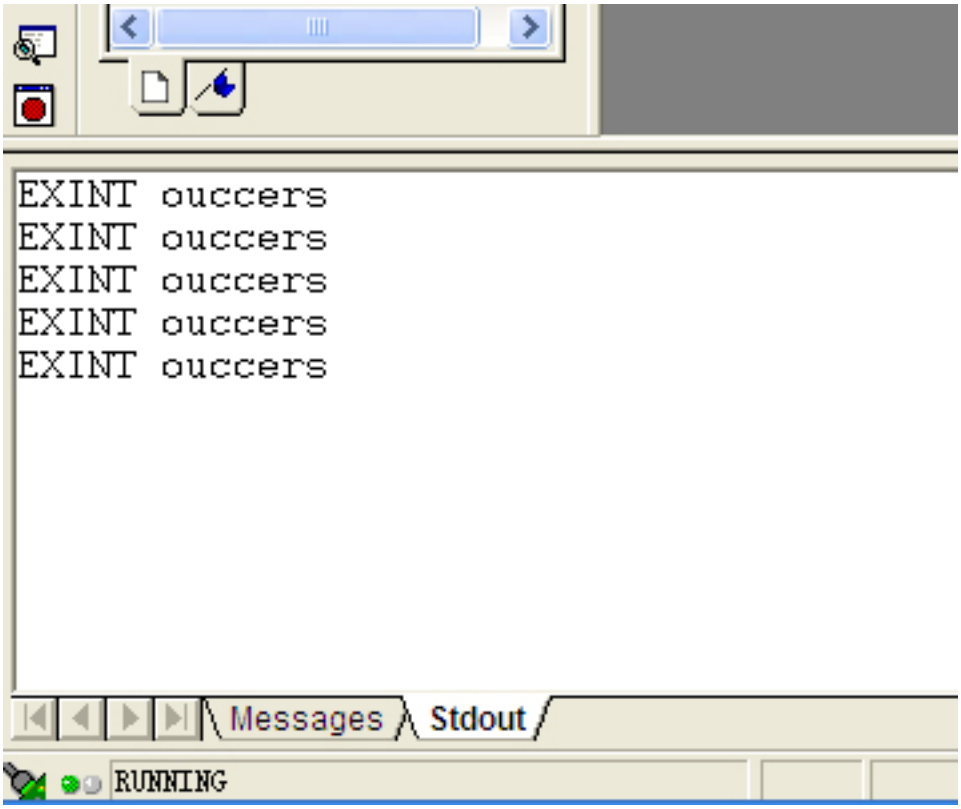
使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509EXINT”文件夹下的“EXINT.PJT”。

（3）点击【File】→ 【Load Program…】，选择“5509 EXINT”文件夹下的“DEBUG”中的“EXINT.OUT”文件，点击【打开】。

（4）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。手按开发板上key5按键，在 CCS 下方的信息栏中即可看到“EXINT ouccers”提示语，提示该外部中断已经产生。如下图所示：



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十四　　AIC23 播音实验

本实验是如何使用 VC5509 控制 AIC23B，发出警报声音的例程。

使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509COEDC”文件夹下的“CODEC.PJT”。

（3）编译程序。

（4）点击【File】→ 【Load Program…】，选择“5509COEDC”文件夹下的“DEBUG”中的“COEDC.OUT”文件，点击【打开】。

(5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。将耳机插入开发板上“PHONE”插孔内，就能听到耳机中发出警报声。

注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十五　　LCD 显示实验

这个例子是如何使用开发板上 LCD 显示汉字的例程。

板子上电前，请先把 LCD 插上。本程序使用于开发板配套的 DM12864M 液晶模块，其他型号的 LCD 请自行更改程序。

使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509LCD”文件夹下的“LCD.PJT”。

（3）加载“5509LCD”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→ 【Load Program…】，选择“5509LCD”文件夹下的“DEBUG”中的“LCD.OUT”文件，点击【打开】。

（5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。观察开发板上 LCD，上面显示“鸿翔电子”字样。

注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十六　　串口通信实验

这个例子是如何使用开发板上的 RS232 接口与 PC 通讯的例程。

板子上电前，请先把标配的 9 针串口线（直连线）插上。

使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509UART”文件夹下的“UART.PJT”。

（3）加载“5509 UART”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→ 【Load Program…】，选择“5509 UART”文件夹下的“DEBUG”中的“UART.OUT”文件，点击【打开】。

（5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。

（6）打开“串口调试助手”（其他串口观察工具也可），将参数设置为“19200bps、无校验位、8 位数据位、1 位停止位”，然后在“发送框”中输入一个数字，点击【发送】，

即可在“接受框”中看到开发板传回来的一模一样的数字。如下图所示。

注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十七　USB2.0 通信实验

这个例子是如何使用 VC5509 中 USB1.1 接口与 PC 机通讯的例程。

使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509USB”文件夹下的“DSK5509_USB.PJT”。

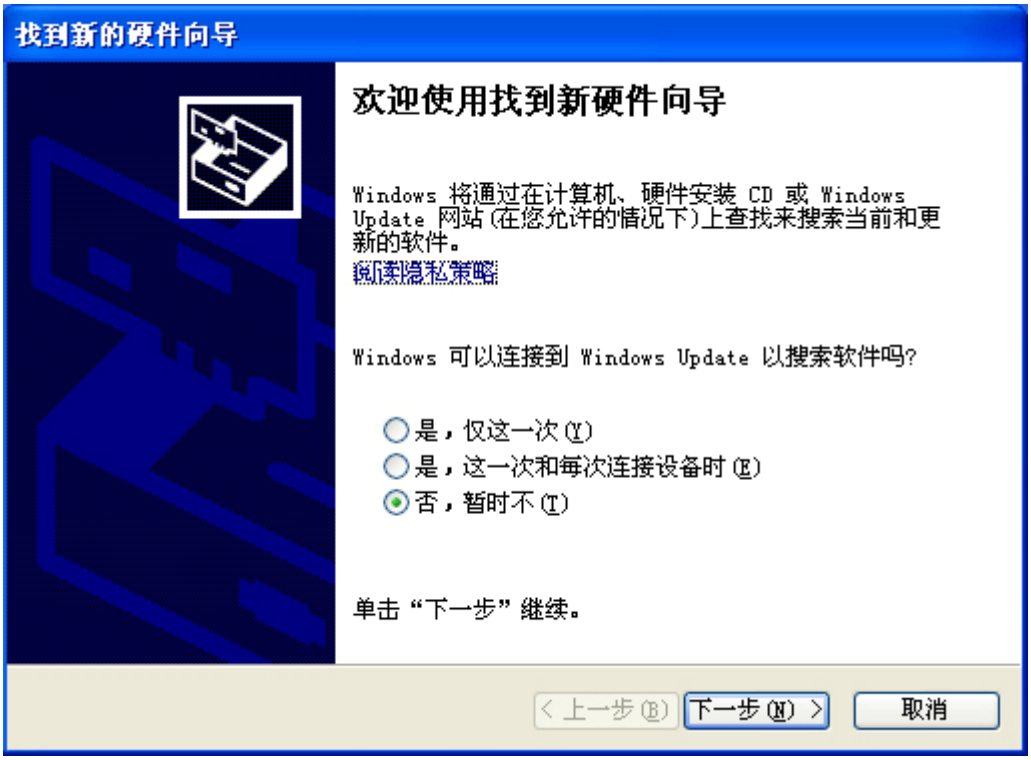
（3）加载“5509USB”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→ 【Load Program…】，选择“5509USB”文件夹下的“DEBUG”中的“DSK5509_USB OUT”文件，点击【打开】。

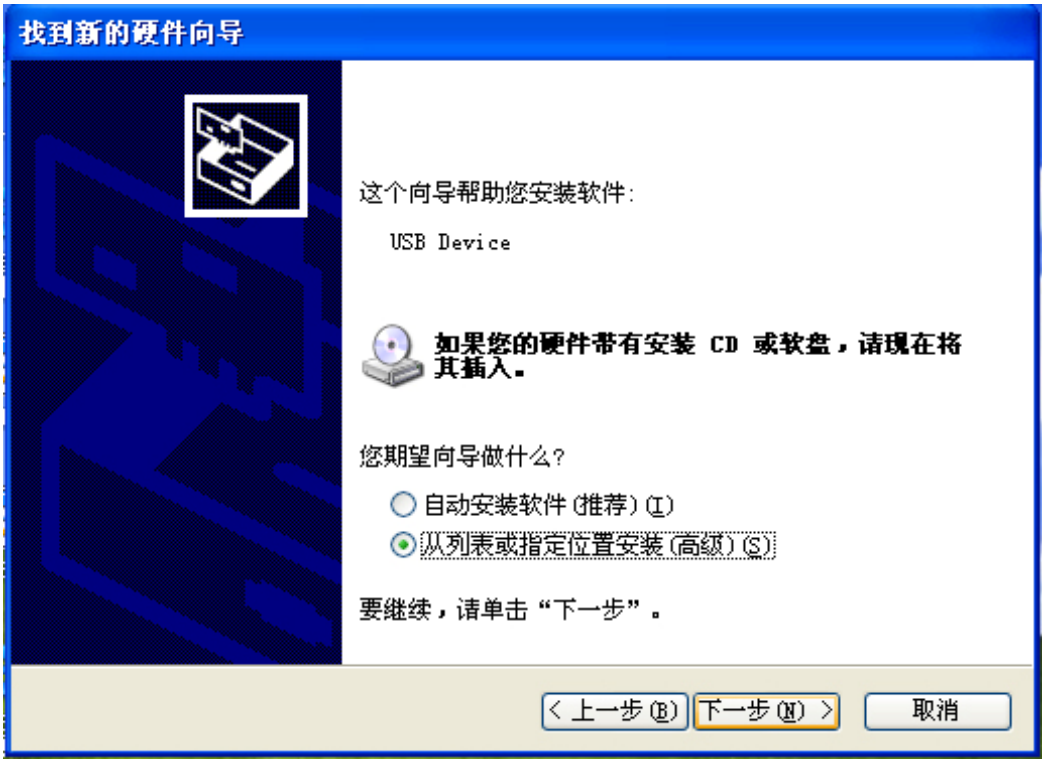
（5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。然后用 USB 延

长线连接开发板上 USB B 型口与电脑的 USB 口，此时，Windows 右下角就会提示【发现新硬件】，如下图所示。

接着会出现如下图所示的【找到新的硬件向导】对话框。该对话框中选【否，暂时不（T）】，然后点击【下一步】。



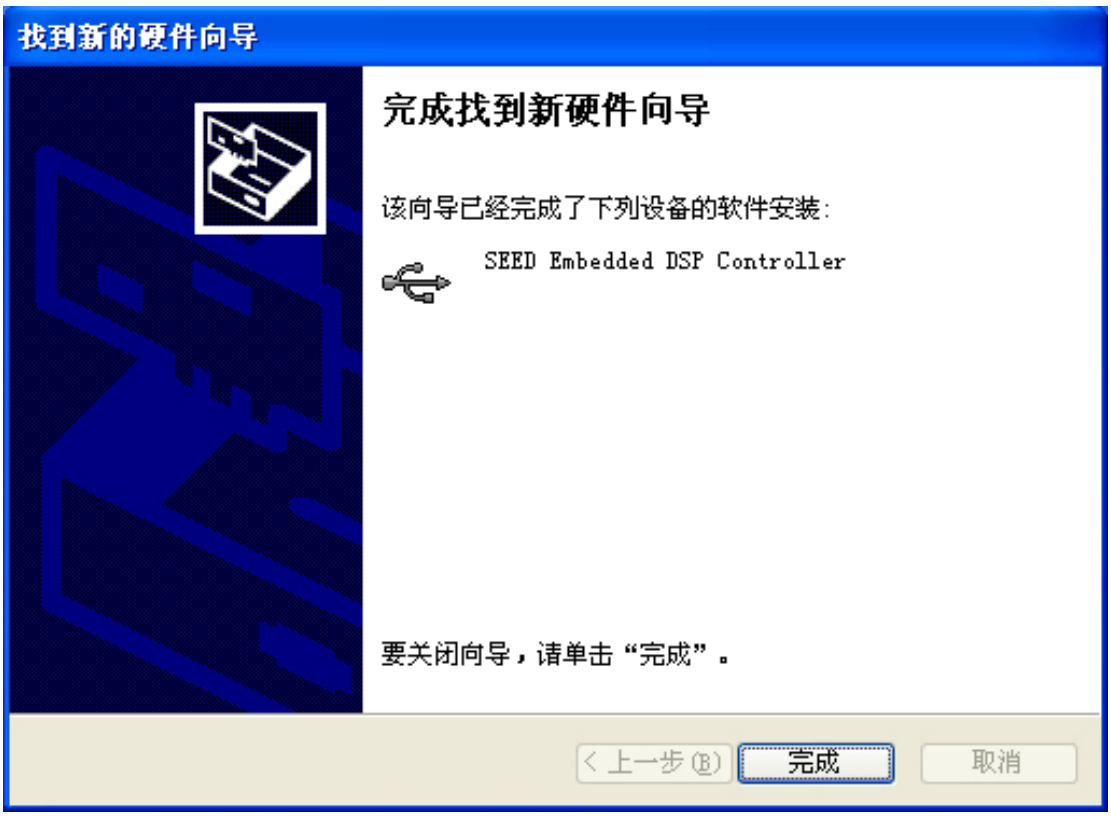
出现如下界面，选【从列表或指定位置安装（高级）】，点击【下一步】。



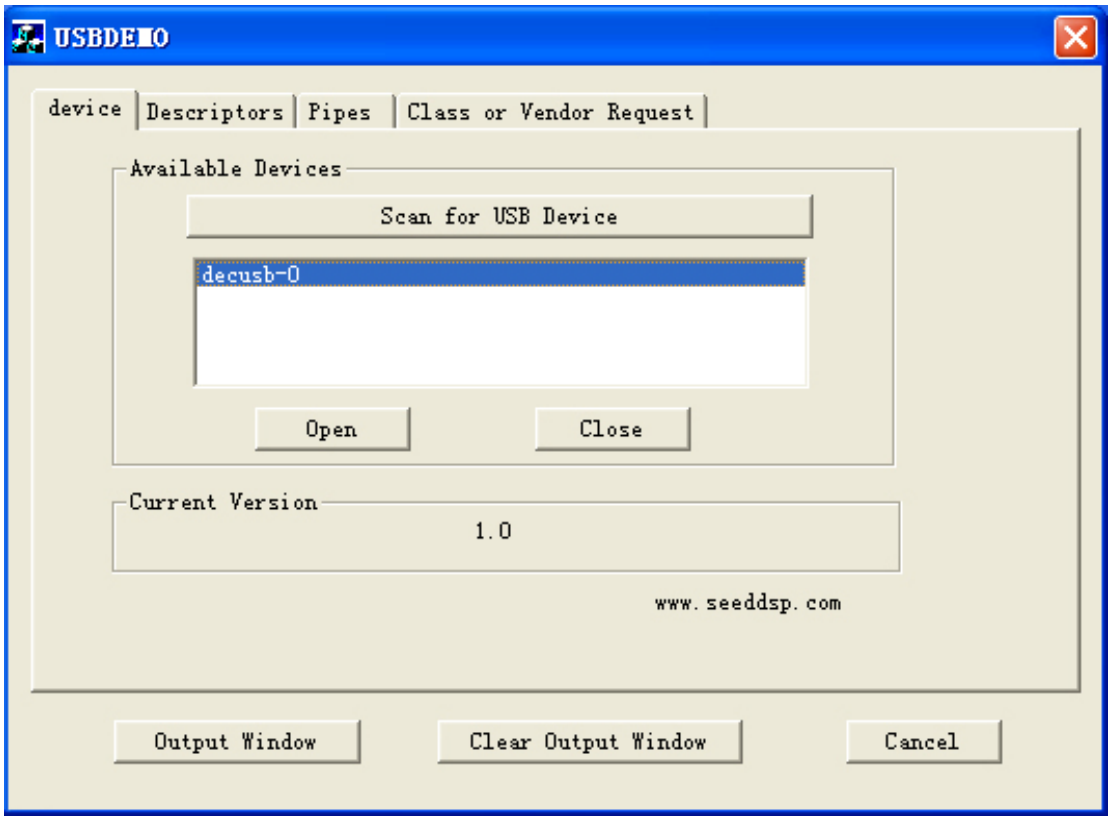
出现如下界面，选择【在搜索中包括这个位置】，然后点击【浏览】，在弹出的文件夹选择框中，指向光盘中的【USB_driver】文件夹，点击【确定】，再点击【下一步】

出现如下界面，如果有如下的界面，则点击【仍然安装】。

直到电脑完成驱动程序的安装，出现如下界面，点击【完成】即可



双击光盘中的“USBDEMO.exe”，打开后如下图所示。点击【Scan for USB Device】，即可发现一个新 USB 设备【decusb-0】，用该上位机程序，即可实现开发板与 PC 通过 USB 来通讯。



注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十八　网络通信实验

这个例子是如何使用 VC5509 通过 8019 实现以太网通讯的例程。

进行该实验前，请安装资料光盘上的 SPYNET 软件。

使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

（2）点击【project】→【Open…】打开“5509IP”文件夹下的“IP.PJT”。

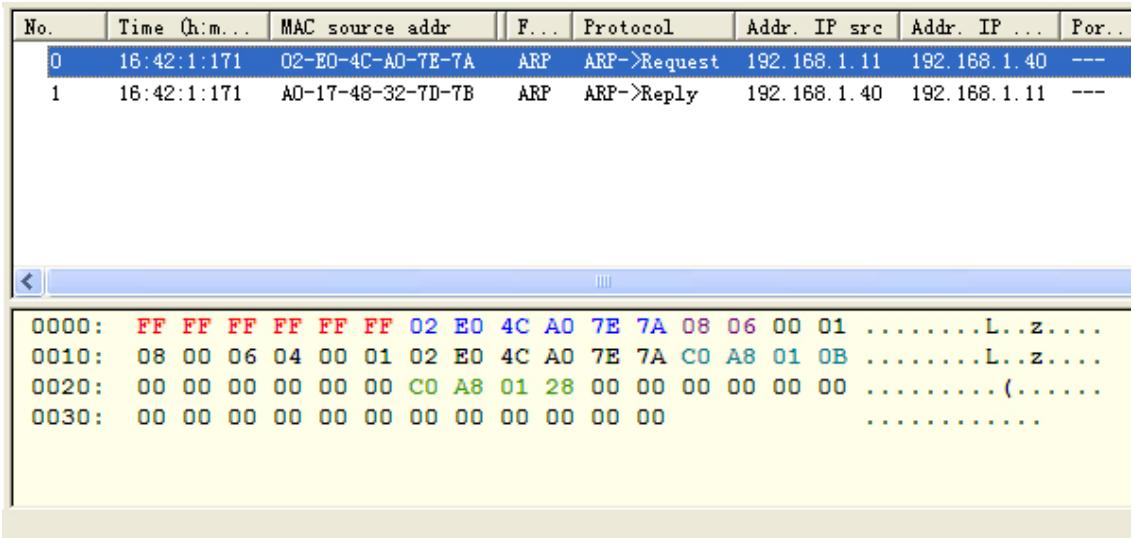
（3）点击【File】→ 【Load Program…】，选择“5509IP”文件夹下的“DEBUG”中的“IP.OUT”文件，点击【打开】。

（4）打开 SPYNET 中的 CAPUTERNET 软件（在开始菜单的【所有程序】中找）。并根据自己的实际网卡进行选择设置。设置完毕后，点击【Start Caputer】，开始监视以太网上的数据包。

（5）设置电脑的 IP 地址为“192.168.1.40”

（6）点击 CCS 中【Debug】→【Run】或左侧快捷键  图标，全速运行。

（7）观察 SPYNET 获取的数据包。这是一个 ARP 请求转换 IP 与物理地址的数据包。如下图所示：



图中有连个数据包，第一个是 ARP 请求，是开发板发给电脑的；第二个是 ARP 回应，

是电脑回应给开发板的。

注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

实验十九　MMC/SD 卡读写实验

这个例子是如何使用 VC5509 中 McBSP 读写 MMC/SD 卡的例程。

使用时，按以下步骤进行：

（1）点击“CCS”，启动 Code Composer Studio 开发环境

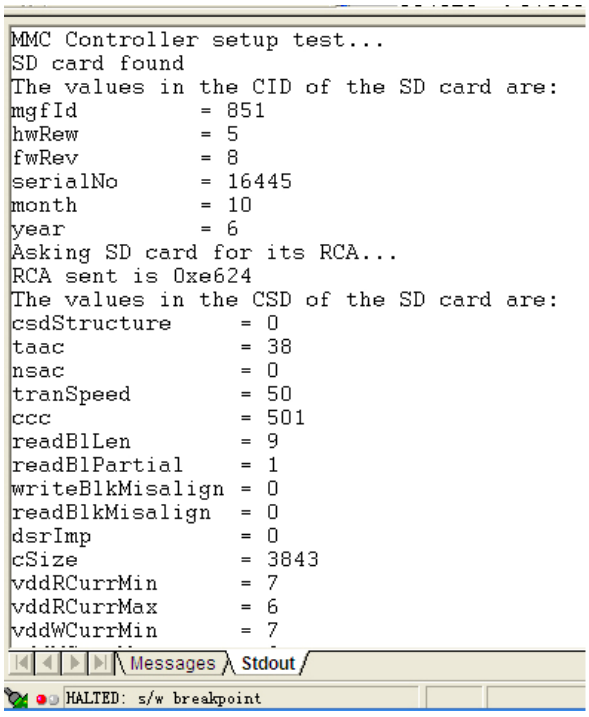
（2）点击【project】→【Open…】打开“5509MMC”文件夹下的“MMC.PJT”。

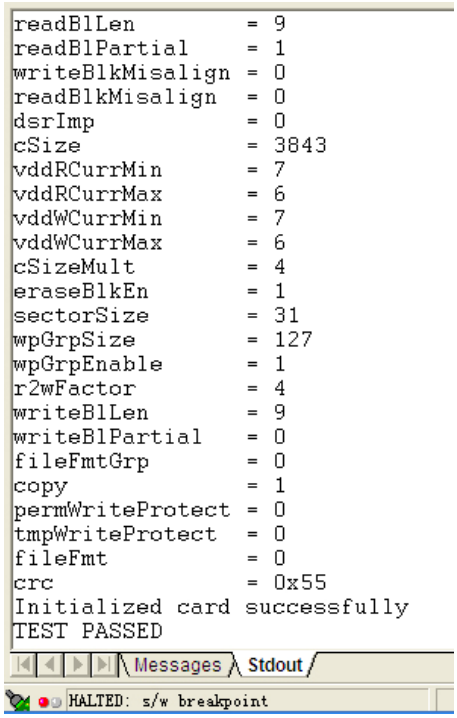
（3）加载“5509MMC”文件夹下的“C5509.GEL”文件（加载方法见本章开头的论述）

（4）点击【File】→ 【Load Program…】，选择“5509MMC”文件夹下的“DEBUG”中的“MMC.OUT”文件，点击【打开】。

（5）点击【Debug】→【Run】或左侧快捷键  图标，全速运行。即可看到 CCS 下方框内输出 MMC 或 SD 卡的信息。如下面的左图所示。

最后，读写成功后，会输出“Initialized card successfully”“TEST PASSED”字样，如下面的有图所示。





注：如果发现程序不能正确运行或程序跑飞，可点击 CCS 中【Debug】→【Reset CPU】，然后重新 Load 程序。如果还是不能正确运行或程序跑飞，可给开发板和仿真器重新上电。

注：做此实验之 MMC/SD 卡上治疗请事先备份，实验完毕，如果 MMC/SD 卡不能打开，请将其格式化一次